Multi-scale Clustering of Frame-to-Frame Correspondences for Motion Segmentation

Ralf Dragon, Bodo Rosenhahn, and Jörn Ostermann

Institut für Informationsverarbeitung
Leibniz Universität Hannover, Germany

Abstract. We present an approach for motion segmentation using independently detected keypoints instead of commonly used tracklets or trajectories. This allows us to establish correspondences over non-consecutive frames, thus we are able to handle multiple object occlusions consistently. On a frame-to-frame level, we extend the classical split-and-merge algorithm for fast and precise motion segmentation. Globally, we cluster multiple of these segmentations of different time scales with an accurate estimation of the number of motions. On the standard benchmarks, our approach performs best in comparison to all algorithms which are able to handle unconstrained missing data. We further show that it works on benchmark data with more than 98% of the input data missing. Finally, the performance is evaluated on a mobile-phone-recorded sequence with multiple objects occluded at the same time.

1 Introduction

Motion is a key element for image understanding. Many animals can easily get fooled by prey standing still for long times. In computer vision, the most-known exploit of motion is to learn 3D shapes of rigid objects by structure-from-motion. Furthermore, recent image segmentation approaches [1,2,3] make use of long-time motion observation as prior for image segmentation. To determine motion, trajectories are established, e.g. from dense optical flow [4] or from tracking feature points like KLT. The trajectories are compared and classified according to their motion, which may be generic 2D or 3D motion or specialized motion templates as used in [5]. For higher discriminativity, it is of advantage to observe long-time trajectories like [6]. However in many real-world applications, trajectories

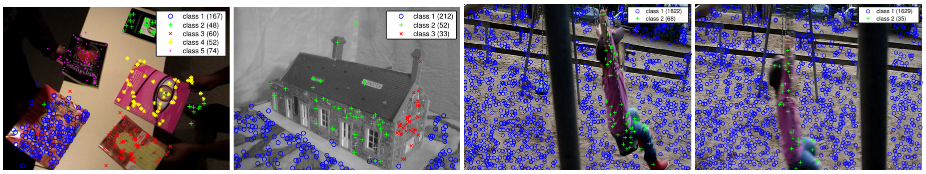

Fig. 1. Results of our multi-scale motion segmentation with automatic class number assignment using SIFT correspondences between pairs of images as input

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part II, LNCS 7573, pp. 445–458, 2012.

are not only incomplete (data is missing at the beginning and at the end) but split by one or several gaps into different parts. Repairing trajectories [7,8] is only tractable as long as the gap is small and it bears the risk of introducing falsely-connected trajectories.

In our approach, we use correspondences from keypoints like SIFT [9] as input for motion segmentation which, e.g., allows learning object appearance from motion. In contrast to *tracked* features, keypoints are detected independently in different frames. On the one hand, comparing keypoint descriptors allows establishing correspondences over missing trajectory elements. On the other hand, although the redetection rate for keypoints in similar images is quite high, it is still far lower than for tracked features. So by gaining the possibility of matching over multiple frames, we loose trajectory length. In this paper, we show that by building a bottom-up frame-to-frame motion segmentation framework on multiple time scales, we can have the advantages of both tracked features *and* keypoints. Due to this, we are able to accurately segment motions of objects which are temporarily or spatially not visible, for example as they are occluded or leave the field of view. Furthermore, our approach is robust to outliers which are explicitly taken into account during the frame-to-frame motion segmentation. Our contributions are

– the motion-split-and-merge (MSAM) algorithm[1] with keypoint correspondences as input, performing fast motion segmentation on a frame-to-frame basis,
– the multi-scale motion clustering (MSMC) approach[1], which combines frame-to-frame-results of multiple frames *and* scales,
– and a metric to robustly and accurately determine the number of motion classes even under very high missing data percentages.

Related Work. Motion segmentation approaches can be characterized using the terms *subspace*- and *affinity*-based. In both methods, a certain time window is observed and trajectories of points are assigned to one of several objects.

In *subspace*-based methods, all trajectory points are combined in a data matrix $\mathbf{W}$. Under the affine camera model, trajectories from rigid objects form linear subspaces in $\mathbf{W}$, which can be extracted using algebraic transformations and rank constraints. [10] shows a closed form solution. However, most approaches cannot handle missing data (e.g. [10,11,12] with good results besides this). To allow incomplete trajectories, the following subspace techniques have been proposed: In [13] the power factorization, which allows the decomposition of matrices with missing data, is used for matrix decomposition of the generalized PCA (GPCA) of $\mathbf{W}$. The agglomerative lossy compression (ALC) approach [14] tries to segment $\mathbf{W}$ by minimizing the number of bits needed to represent $\mathbf{W}$. However reconstructing missing data is only possible as long as at least one complete trajectory is available. As it can be seen in Fig. 4, such an assumption does not hold here since no object is visible throughout the whole sequence.

[1] Code available on `www.tnt.uni-hannover.de/staff/dragon`

Our approach belongs to *affinity*-based methods. Here, pair-wise affinities between trajectories are computed. They form the affinity matrix $\mathbf{A}$ which is analyzed during a final clustering step. In contrast to *subspace*-based methods, the affinity metric does not depend on complete data. [4] use spectral clustering [15] with affinities from spatial trajectory distances combined with their similarity of translational motion. [16] propose affine motion similarity as basis for affinities which are clustered by J-linkage [17]. Both approaches build $\mathbf{A}$ from motions of consecutive frames only. Thus, slowly-drifting motions still have high affinity. This effect is solved by spatial terms in $\mathbf{A}$, but only if the trajectory density is homogeneous and if the trajectories have high overlap. In [7] the trajectory features speed and direction are decomposed using non-negative matrix factorization. The resulting weights are used as affinities for spectral clustering. However, the factorization cannot handle noisy data or outliers.

Outstanding results were received by combining both *subspace*- and *affinity*-based methods: In [18], a sparse representation is estimated from $\mathbf{W}$ and used as affinity for spectral clustering. However, justification for handling missing data is only given as long as there is at least one trajectory with complete data. Despite this, the computational complexity for the sparse representation does not allow a high number of trajectories (> 6000 in Section 4.3). In our approach, we also combine the advantages of *affinity*-based methods with a long-term optimization like in *subspace*-methods. But our multi-scale motion clustering has low complexity and no constraints how the missing data is formed.

The rest of this paper is structured as follows: In Section 2, we explain our frame-to-frame motion-split-and-merge algorithm (MSAM). In Section 3, we derive our multi-scale motion clustering (MSMC) approach which combines frame-to-frame segmentations of multiple frames and scales. Section 4 shows experimental results and in Section 5, we give a conclusion.

2 Frame-to-Frame Motion Segmentation

A set $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots\}$ of given point correspondences $\boldsymbol{y}_i$ between two frames is to be partitioned into n subsets $\mathcal{F}_k$, called segments here, s.t. $\mathcal{F}_k$ and underlying motion coincide. Furthermore, the number of segments n is to be estimated. Let $\boldsymbol{y}_i = (\boldsymbol{x}_i^{(1)\mathsf{T}}, \boldsymbol{x}_i^{(2)\mathsf{T}})^\mathsf{T}$ be the data vector consisting of the corresponding points $\boldsymbol{x}_i^{(1)}$ and $\boldsymbol{x}_i^{(2)}$. The motion of $\mathcal{F}_k$ is parametrized by the vector $\boldsymbol{p}_k$, in this paper the 8 parameters of a homography.

Since block-wise segmentation for video coding or motion segmentation of dense optical flow denote different problems, there is only [19] as application tackling a similar problem. It consists of J-linkage and many post-processing steps preventing fast runtimes. Since the segmentation is carried out very often in our global framework, we introduce MSAM as fast and precise approach.

2.1 Motion-Split-And-Merge (MSAM)

The inspiration of our approach is the classical top-down split and merge algorithm [20, pp 96–104] for image segmentation, where image segments are split until they are consistent and then merged with neighboring segments until convergence. *Consistency* in our case means that all correspondences $\boldsymbol{y}_i$ from a segment $\mathcal{F}_k$ are inlier according to

$$d^2(\boldsymbol{y}_i \mid \boldsymbol{p}_k) < \epsilon^2 , \tag{1}$$

where $d^2(\boldsymbol{y}_i \mid \boldsymbol{p}_k)$ is the model distance metric for the data $\boldsymbol{y}_i$ using the model parameters $\boldsymbol{p}_k$ of $\mathcal{F}_k$, here the symmetric squared error. ϵ^2 is a threshold in this metric. However, motion parameters $\boldsymbol{p}_k$ have to be estimated from $\mathcal{F}_k$ which is problematic if $\mathcal{F}_k$ consists of multiple underlying models. Furthermore, combining *neighboring* segments in terms of similar parameters will not work if one of the segments is too small for the estimation step. Thus, we formulate motion-split-and-merge in a way we avoid too much splitting. It consists of advanced split and merge operations and explicit outlier handling.

Initially, we assume all correspondences to be in one segment. At each iteration, the following operations are performed:

1. Split segments: For each segment $\mathcal{F}_k$, RANSAC [21] estimates parameters $\boldsymbol{p}_k$. Outliers according to Eq. (1) are assigned to an outlier segment. If however the inlier ratio is smaller than θ_s, we believe that RANSAC could not estimate a valid motion because of too much perturbation from outliers or from multiple models inside the segment. Thus, we split the segment into two by the approach explained in Section 2.3.
2. Merge segments: For each pair of segments $(\mathcal{F}_k, \mathcal{F}_l)$, the inlier ratio τ of the smaller segment $\mathcal{F}_l$ is determined using parameters $\boldsymbol{p}_k$. The intention is that the parameters of the bigger segment are more stable and accurate. Besides, the inliers of the smaller segment can be computed faster. Like in regular split-and-merge, the segments are merged if τ is bigger than a threshold θ_m.
3. Split outlier segment: All outliers are put into one segment which is split like in step 1. If one of both parts contains enough inliers, it is added as a new segment.
4. Merge outlier segment: Each correspondence $\boldsymbol{y}_i$ from the outlier segment is assigned to a segment $\mathcal{F}_k$ if it is inlier according to its parameters $\boldsymbol{p}_k$.

The algorithm ends if no operation was performed during an iteration. Please note that by the outlier handling, we cannot guarantee convergence but to our experience, MSAM converges quickly if the underlying motions are parameterizable under the given ϵ. If not, MSAM is stopped after a cutoff iteration count experimentally found in Section 4.1. In order to disallow repeated splits and merges, a neccessary constraint for θ_s and θ_m is

$$1 + \theta_m \geq 2 \cdot \theta_s . \tag{2}$$

A high θ_s allows RANSAC with only few samplings in step 1 (cf. Eq. (7)), but on the other hand may result in too small segments from many splits and few merges by θ_m being too high. Thus, optimal thresholds depend on outlier percentages and the desired granularity of the segmentation. Since an over-segmentation will not harm the on-top multi-frame motion clustering, we empirically found

$$\theta_m = 0.9, \quad \theta_s = 0.95 \tag{3}$$

as biggest parameters usable for SIFT and KLT correspondences.

2.2 J-Linkage

As the splitting step from Section 2.3 is based on concepts used in J-linkage [17], we briefly repeat it using our notation. J-linkage belongs to the field of model selection, where data vectors $\boldsymbol{y}$ (not necessarily correspondences) are to be assigned to one of several underlying segments having different parameters $\boldsymbol{p}_k$. J-linkage consists of a sampling and a clustering step. In the sampling step, M model parameter vectors $\boldsymbol{p}$ are estimated from M varying randomly-selected minimal data sets. To increase the probability of drawing samples with the same underlying parameters, samples with similar $\boldsymbol{y}$ are drawn with higher probability.

After the sampling, for each correspondence $\boldsymbol{y}_i$ the set of its preferred parameters $\mathcal{P}(\boldsymbol{y}_i)$, or short $\mathcal{P}_i$, is found containing all parameter vectors $\boldsymbol{p}$ to which it is inlier (cf. Eq. (1)):

$$\boldsymbol{p} \in \mathcal{P}_i \quad \Leftrightarrow \quad d(\boldsymbol{y}_i \mid \boldsymbol{p}) < \epsilon\,. \tag{4}$$

In the clustering step segments are found using bottom-up agglomerative clustering, starting with many segments $\mathcal{F}_k = \{\boldsymbol{y}_k\}$ each containing one data vector. For further explanation, let the model preference set $\mathcal{M}_k$ of a segment $\mathcal{F}_k$ contain the common preferred parameter vectors of all its data

$$\mathcal{M}_k := \bigcap_{\boldsymbol{y}_i \in \mathcal{F}_k} \mathcal{P}(\boldsymbol{y}_i)\,. \tag{5}$$

In J-linkage, iteratively the two segments $\mathcal{F}_k$ and $\mathcal{F}_l$ with lowest Jaccard distance $J(\mathcal{M}_k, \mathcal{M}_l)$ between their model preference sets are combined. The Jaccard distance for two sets $\mathcal{A}$ and $\mathcal{B}$ is defined as

$$J(\mathcal{A}, \mathcal{B}) = \begin{cases} \frac{|\mathcal{A} \cup \mathcal{B}| - |\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|} & \mathcal{A} \cup B \neq \emptyset \\ 1 & \text{otherwise}\,. \end{cases} \tag{6}$$

The second case may occur if both $\mathcal{F}_k$ and $\mathcal{F}_l$ have empty preference sets since they are outliers to all estimated parameter vectors. The clustering ends if all pairs $(\mathcal{F}_k, \mathcal{F}_l)$ have disjoint $\mathcal{M}_k$ and $\mathcal{M}_l$ and thus $J(\mathcal{M}_k, \mathcal{M}_l) = 1$. Last, small clusters are combined to an outlier segment.

Although the clustering stop criterion sounds reasonable, the agglomerative clustering suffers from local optima by combining wrong segments (e.g. of ambiguous correspondences or outliers). This leads to over-segmentation as these partitions cannot be merged because the model preference sets $\mathcal{M}_k$ become too sparse. Furthermore, the parameters for the clustering are inaccurate as they have always been found from local minimal sampling sets. These problems could be overcome (cf. Fig. 2) by a motion validation which is performed by our MSAM approach. Thus the idea is to use a simplified and sped up version of J-linkage in step 1 of MSAM which is described in the following.

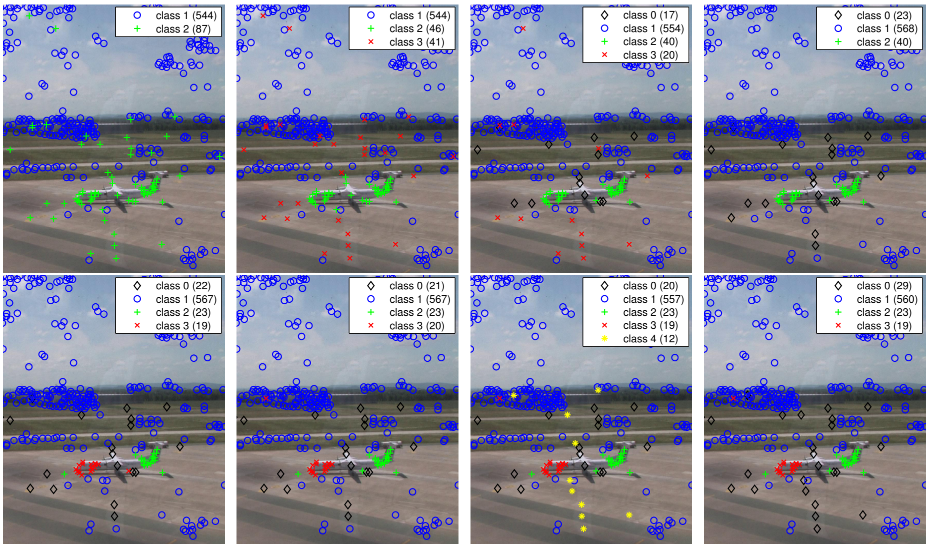

Fig. 2. The segmentation of a turning airplane using (upper row) our MSAM approach after 1, 2, 4 and 6 of 6 iterations, and using (lower row) J-linkage [17] with optimized parameters and 2500, 5000, 7500 and 10000 samplings. The motion model and the outlier threshold from Eq. (4) are the same. Using MSAM, the airplane is correctly segmented after 6 iterations. Motion class 3 is split in iterations 2 and 4. One of the split parts gets merged with the background such that the outliers (class 0) are increasingly detected. After convergence, all 23 outliers were detected correctly and the airplane as well as the background class are motion consistent. Using J-linkage, the correspondences are still oversegmented and not all outliers are detected. More iterations seem not to give better results.

2.3 MSAM Split Operation

The goal of this operation is to split the correspondences of a non-consistent segment $\mathcal{F}_k$ into motion consistent partitions. As further splits as well as merges may follow this operation, we may assume that we have to split $\mathcal{F}_k$ into two parts. This allows sampling preferred parameters $\mathcal{P}_i$ using a far lower number of parameter estimation sets M as in J-linkage. Here we chose $M = 10$. Furthermore, we enhance the parameter vector $\boldsymbol{p}$ in inlier count and precision by using the best RANSAC result of r samplings. r is chosen such that the parameters are estimated from true inliers with more than $p = 50\%$, assuming we have at least $w = 50\%$ model inliers. From [21], we know that

$$r \geq \frac{\log(1-p)}{\log(1-w^s)} \approx 11\,, \tag{7}$$

where $s = 4$ is the size of a minimal sampling set.

After this sampling step, the Jaccard distance $J(\mathcal{P}_i, \mathcal{P}_j)$ is computed according to Eq. (6), in contrast to J-linkage between the preferred parameter sets of *all* pairs of correspondences $\boldsymbol{y}_i, \boldsymbol{y}_j \in \mathcal{F}_k$. In order to include spatial constraints, we combine the Jaccard distance with the Mahalanobis distance M_k:

$$M_k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\intercal \mathbf{C}_{\boldsymbol{x}}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)} \quad \wedge \quad \boldsymbol{x} \in \mathcal{F}_k \, , \tag{8}$$

where $\boldsymbol{x}$ is wlog. the destination point of correspondence $\boldsymbol{y}$ and $\mathbf{C}_{\boldsymbol{x}}$ is the covariance matrix of $\boldsymbol{x}$. To combine J and M_k, the affinity matrix $\mathbf{A}$ is defined:

$$A_{i,j} = 1 - J(\mathcal{P}_i, \mathcal{P}_j) \cdot \min\{M_k(\boldsymbol{x}_i, \boldsymbol{x}_j), 1.0\} \, . \tag{9}$$

The distances J and M_k are combined multiplicatively in order to cluster correspondences which are spatially *or* by common model parameters near to each other. To give both the same weight, M is saturated to the value range $[0, 1]$. By this, clustering of isolated correspondences randomly moving together is inhibited while elongated or discontinuous clusters remain possible. Finally, the columns of $\mathbf{A}$ and by this $\mathcal{F}_k$ is split into two segments using k-Means.

2.4 Runtime and Precision Considerations

Comparing our motion segmentation with J-linkage as strongest other approach, there are three main reasons for speedups and three for enhanced precision which justify using MSAM for frame-to-frame motion segmentation: In total, we have to perform less samplings for parameter estimation, most samplings are used for splitting classes containing only a fraction of all correspondences, and the clustering method may be simpler since further parameter validations are possible. On our equipment (cf. Section 4), the segmentations of 631 correspondences in Fig. 2 took 0.7 s for MSAM, where J-linkage with just 2500 samplings took 20.1 s (just the clustering took 16.6 s), both on compiler-optimized C code.

Since MSAM's parameters are estimated using *all* correspondences from a segment (and not minimal sampling sets), they are more precise. Second, because J-linkage draws samples spatially close to each other, its parameters become prone to imprecise feature localization, while MSAM incorporates spatial knowledge during the clustering when parameters are already estimated. Furthermore, because of the explicit outlier handling, all segments are motion consistent if MSAM terminates regularly. As it is displayed in Fig. 2, MSAM finds the true solution whereas J-linkage with the same outlier distance and optimized parameters performs an over-segmentation containing outliers in all classes.

3 Global Multi-scale Optimization

3.1 Global Motion Segmentation Using Successive Frames

To combine the results from multiple frame-to-frame motion segmentations in a time-consistent way, linked frame-to-frame inlier correspondences $(\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)})$, $(\boldsymbol{x}_i^{(2)}, \boldsymbol{x}_i^{(3)})$, ... are assigned to trajectories T_i. In contrast to the clustering in Section 2, we now cluster the set of all T_i according to multiple frame-to-frame

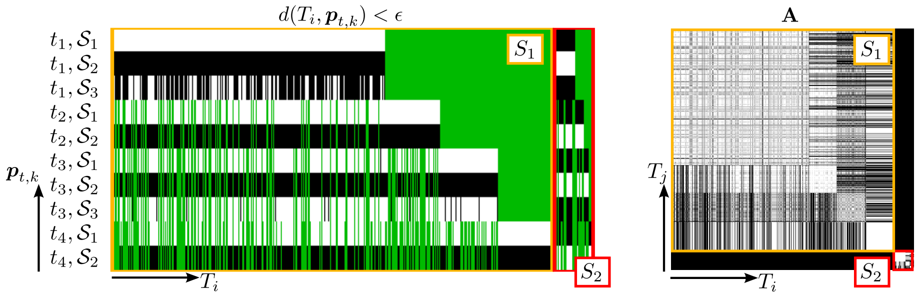

Fig. 3. Left: Preferred motion parameters $\boldsymbol{p}_{t,k}$ of trajectory T_i, clustered by motions $\mathcal{S}_1$ (yellow) and $\mathcal{S}_2$ (red). White denotes inliers, black outliers and green missing data (no observation of T_i at frame t). Right: Corresponding affinity matrix $\mathbf{A}$ of the affinity between trajectories T_i and T_j.

motions. Let $\mathcal{F}_{t,k}$ be a segment of a frame-to-frame segmentation of correspondences $\boldsymbol{y}^{(t)} = (\boldsymbol{x}^{(t)\intercal}, \boldsymbol{x}^{(t+1)\intercal})^\intercal$, and accordingly let $\boldsymbol{p}_{t,k}$ be its parameter vector.

Similar to Eq. (4), we define the preferred parameter set $\mathcal{H}(T_i)$, or short $\mathcal{H}_i$, of trajectory T_i containing the parameters $\boldsymbol{p}_{t,k}$ from different frames t and different motions k, to which T_i is inlier (cf. Fig. 3):

$$\boldsymbol{p}_{t,k} \in \mathcal{H}_i \quad \Leftrightarrow \quad \boldsymbol{y}_i^{(t)} \in T_i \quad \wedge \quad d(\boldsymbol{y}_i^{(t)} \mid \boldsymbol{p}_{t,k}) < \epsilon \,. \tag{10}$$

To segment all trajectories into n spatio-temporal motion clusters $\mathcal{S}_u$, we use spectral clustering [15]. Its affinity matrix $\mathbf{A}$ which contains pairwise affinities $A_{i,j}$ between two trajectories T_i and T_j, is defined using Eq. (10):

$$A_{i,j} = 1 - J(\mathcal{H}_i, \mathcal{H}_j) \,. \tag{11}$$

By this, we allow clustering trajectories with missing data without any constraints how the missing data is formed.

3.2 Multi-scale Motion Clustering (MSMC)

In contrast to the approaches of [4,16], our presented trajectory clustering allows combining independent frame-to-frame segmentations over arbitrary time scales h_Δ. By this, slow or randomly coinciding motions can be segmented. To combine the results of different scales, linked inlier correspondences are merged and treated as part of the same trajectory. The indices t, k from Section 3.1 become t, Δ, k denoting motion k from frame t to frame $t + \Delta$. Eq. (10) becomes the definition of the preferred motion parameter set $\mathcal{H}'(T_i)$ over multiple scales:

$$\boldsymbol{p}_{t,\Delta,k} \in \mathcal{H}'_i \quad \Leftrightarrow \quad \boldsymbol{y}_i^{(t,\Delta)} \in T_i \;\; \wedge \;\; d(\boldsymbol{y}_i^{(t,\Delta)} \mid \boldsymbol{p}_{t,\Delta,k}) < \epsilon \,, \tag{12}$$

where $\boldsymbol{y}_i^{(t,\Delta)} = (\boldsymbol{x}_i^{(t)\intercal}, \boldsymbol{x}_i^{(t+\Delta)\intercal})^\intercal$. Using $\mathcal{H}'_i$ instead of $\mathcal{H}_i$ for the affinities of the spectral clustering of trajectories T_i in Eq. (11), correspondences over *all scales* and *all frames* are clustered in one step (cf. Fig. 4).

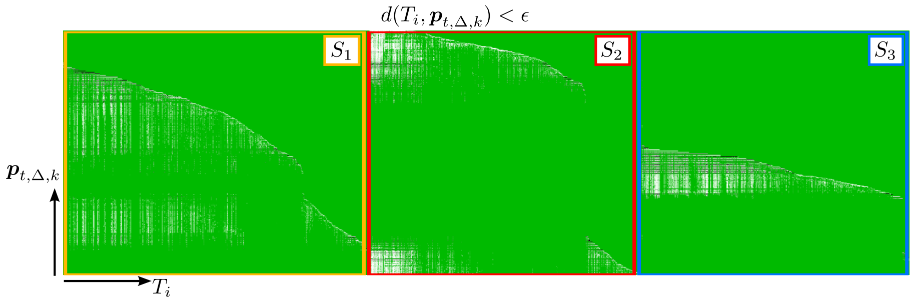

Fig. 4. Preferred motion parameters $\boldsymbol{p}_{t,\Delta,k}$ of 6795 trajectories and 768 frame-to-frame motions from the bookstack sequence (Section 4.3). Several objects are occluded at the same time. White denotes inliers, black outliers and green missing data (98.5%), e.g. due to occlusion. $\mathcal{S}_2$ corresponds to the lowest book and $\mathcal{S}_3$ to the topmost (cf. Fig. 8).

3.3 Estimation of the Number of Motion Classes

In the following we present a simple yet powerful method to determine the number of motions n by comparing the results of clusterings having different motion counts ξ. For each ξ, the $\xi \times \xi$ class similarity matrix $\mathbf{B}(\xi)$ is established. Element $B_{u,v}(\xi)$ contains the sum of affinities of all trajectories $T_i \in \mathcal{S}_u$ and $T_j \in \mathcal{S}_v$, normalized by the total overlap between $\mathcal{S}_u$ and $\mathcal{S}_v$:

$$B_{u,v} = \frac{\sum_{T_i \in \mathcal{S}_u} \sum_{T_j \in \mathcal{S}_v} (1 - J(\mathcal{H}'(T_i), \mathcal{H}'(T_j)))}{\sum_{T_i \in \mathcal{S}_u} \sum_{T_j \in \mathcal{S}_v} |T_i \cap T_j|} \,. \tag{13}$$

If there is no overlap between trajectories from $\mathcal{S}_u$ and $\mathcal{S}_v$, we set $B_{u,v} = 0$.

Since different motions $\mathcal{S}_u \neq \mathcal{S}_v$ should have a low similarity $B_{u,v}$ and the self-similarity $B_{u,u}$ should be high, we find the optimal number of classes n by a worst case minimization:

$$n = \arg\min_{\xi} \left(\max_{u \neq v} B_{u,v}(\xi) - \min_{u} B_{u,u}(\xi) \right) \,. \tag{14}$$

Minimizing only the first $B_{u,v}$- or the second $B_{u,u}$-term results in an under- or oversegmentation, respectively. As we will show in the experiments, the combination of both terms result in a quite accurate and robust estimation of n.

4 Experiments

4.1 Complete Data

We start analyzing the segmentation performance using complete trajectories from the Hopkins 155 benchmark [22]. We use the non-synthetic *Articulated* and the *Traffic* data set, each separated into two- and three-motion sequences. In all experiments, we use three scales h_1 (frame-to-frame tracking), h_5 (to close gaps)

Table 1. Classification error rates and recall in the Hopkins data set without missing data. The CPU times are collected from different papers with comparable systems.

Approach	ALC[14]	GPCA[22]	NNMF[7]	VLS[16]	**MSMC**
Articulated, 2 motions, 11 sequences					
Error	10.70%	2.88%	10.00%	5.38%	6.03%
Recall	1	1	1	1	0.963
Articulated, 3 motions, 2 sequences					
Error	21.08%	16.85%	15.00%	20.41%	8.27%
Recall	1	1	1	1	0.974
Traffic, 2 motions, 31 sequences					
Error	1.75%	1.41%	0.10%	1.92%	0.66%
Recall	1	1	1	1	0.979
Traffic, 3 motions, 7 sequences					
Error	8.86%	19.83%	0.10%	4.89%	0.17%
Recall	1	1	1	1	0.987
All 51 sequences					
Error	**5.41%**	**4.86%**	**2.82%**	**3.80%**	**2.05%**
Recall	1	1	1	1	0.977
CPU time/sequence	261.3s	0.3s	3.0s	5.7s	13.8s

and h_{25} (long-time motion analysis). To demonstrate MSAM converges even under very small outlier thresholds ϵ (cf. Eq. (4)), it was set to $\epsilon = 0.5$ pel. First, we use the provided number of classes for the segmentation. Since RANSAC and k-Means introduce a fluctuation of the results, we average over 10 repetitions.

The convergence of MSAM over all sequences and repetitions is displayed in Fig. 7 (b): More than 90% of all segmentations converge with less than 10, and 99% with less than 20 iterations. The rest is likely to contain motions not parameterizable by homographies, so we terminate MSAM after 20 iterations.

Next we compare our MSMC to all existing approaches which can handle missing data without completeness constraints (which [14,18] have, cf. Section 1): subspace-based GPCA ([13], results from [22]), and the affinity-based approaches from [7] (NNMF) and [16] (various life span – VLS). We further add ALC [14] since we will refer to it in Section 4.2. Traditionally, all trajectory elements are assigned to motion segments and classification errors are reported. As our approach may not only result in such true (tp) and false positives (fp), but also in false negatives (fn), we evaluate

$$\text{precision} = {}^{\text{tp}}/_{\text{tp+fp}}, \quad \text{and recall} = {}^{\text{tp}}/_{\text{tp+fn}}\,. \tag{15}$$

error = 1 − precision equals the usual benchmark performance metric [22].

The results in terms of average error rate and recall are displayed in Table 1. It can be observed that we receive the best overall error rate at a recall very close to 1 and reasonable runtime on a 3.0 GHz quadcore standard desktop PC. Using the metric from Section 3.3, the number of motions could be estimated with an accuracy of 78.4%, which is the best result reported so far: [23] received an accuracy of 63.4% and [24] 74.8% (including the synthetic sequences).

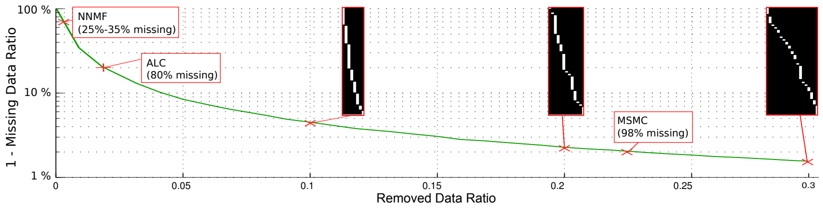

Fig. 5. Simulation how *removed* data affects *missing* data. The three binary images are examples of the result of one trajectory getting split because of gaps from 10%, 20%, and 30% *removed* data. The images show if data is known (white) or missing (black) for different resulting trajectories (columns) over time (rows).

4.2 Missing Data

Starting with the complete data, we randomly remove data from the trajectories as it occurs when features are not detected. By such gaps, trajectories become separated. For realistic evaluation, a trajectory with a gap is treated as two different disjoint trajectories, one ending before the gap and one starting after it. We use the term *removed data* for the amount of gaps introduced and *missing data* for the amount of resulting missing trajectory data given to the algorithm. When trajectory points are only removed at the ends as in [7], both terms coincide. However if, e.g., one trajectory element is removed in the middle of f frames, the removed data ratio is $1/f$ whereas the missing data ratio is $\rho = 0.5 + 2/f$ for the two resulting trajectories (cf. Fig. 5). By this, the clustering of trajectories becomes much more complex. Our approach reduces the complexity by merging trajectories on different scales as long as the motion segmentation does not fail (e.g. because of too little available data). Using this procedure, we randomly *remove* up to 30% data, which results in up to 98, 5% *missing* data. As a comparison: In the trajectory recovery of [14] (ALC), by removing complete trajectories, up to 80% data was *missing* which corresponds to $\rho \approx 2\%$ *removed* data here. The masks used to simulate missing data in [7] (NNMF) contained 25%-35% missing data at beginning and ends of trajectories corresponding to $\rho < 1\%$ (cf. Fig. 5). We evaluate precision, recall and accuracy of class number estimations over removed data using the setup from Section 4.1.

Fig. 6 displays the results with [7] as comparison. It can be observed that our approach is very robust with respect to missing data in terms of precision, recall and class estimation accuracy. The outcomes are reasonable even if more than 20% of the data is *removed*, corresponding more than 98% *missing* data. This justifies that we can use MSMC with keypoint correspondences as input.

4.3 Multi-scale SIFT Motion Clustering

In this experiment, keypoint correspondences from SIFT features [9] are used for motion segmentation. For fast computation of features and correspondences, we use the GPU approach from [25]. We demonstrate our multi-scale approach

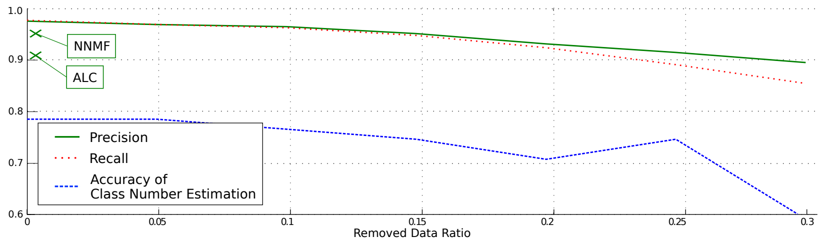

Fig. 6. Average precision, recall and class number estimation accuracy from Section 4.2 over the ratio of removed data. The precisions from ALC and NNMF are extracted from [7].

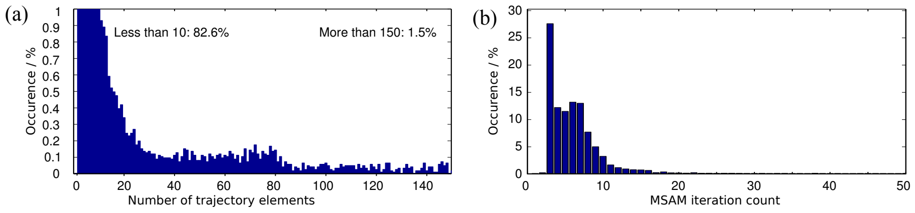

Fig. 7. (a) Histogram of the number of known trajectory elements for the bookstack sequence (Section 4.3). (b) Histogram of MSAM iteration count of all frames, scales and sequences of the experiment in Section 4.1.

on the bookstack video sequence (225 frames, $840 \times 480\,\text{pel}^2$, Fig. 8) which was taken with a regular mobile phone camera. Thus, the captured video contains motion blur and artifacts from compression and rolling shutter. The books occlude each other in the first half of the sequence and get uncovered later. The background intentionally was left blank to give the algorithm no information about the motion of the lowest book. We used three objects to demonstrate that multiple objects may become occluded at the same time. If only one object is occluded and the number of motions is given, an occlusion would be solved by assigning disjoint classes the same motion. With the following exceptions, the parameters remained the same as in previous experiments. To be able to segment consistently through the long-time occlusion, we added scales h_{100} and h_{200}. Further, to reduce computational complexity because of too many frame-to-frame motions segments, the outlier threshold from Eq. (4) was set to $\epsilon = 3\,\text{pel}$. As baseline for comparison, we input the sequence into the approach of [4] which is based on dense trajectories from optical flow.

The preferred parameter sets for 768 frame-to-frame motion segments and 6795 trajectories, containing 67103 SIFT inliers, is displayed in Fig. 4, the histogram of trajectory lengths in Fig. 7 (a), and images with classified features are shown in Fig. 8. It can be observed that MSMC allows very high precision under full occlusions, high image perturbation and very high missing trajectory

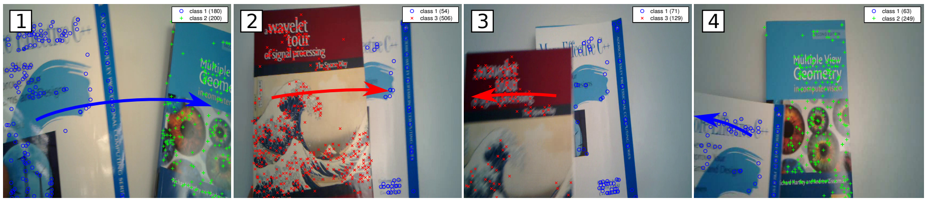

Fig. 8. Images from the bookstack sequence with the segmentation result: The green-labeled book becomes occluded by the blue-labeled book at $t = 1$ which in turn becomes occluded by the red-labeled book at $t = 2$. The books are assigned consistently again.

data rates (here 98.5%). The number of motions was correctly estimated as 3, corresponding to the three books. The precision, measured every 10 frames, is at 98.2%. The baseline approach of course fails in re-assigning the books. But even when taking only the stacking of the books into account, it results in severe oversegmentation such that the precision is only 61.9%. Further, with a frame rate of 4.1 s for motion segmentation plus 30.2 s for the optical flow, the baseline approach is by far slower than MSMC with an overall rate (including SIFT extraction and matching) of 1.1 s per frame. We think that by introducing mechanisms to feed back the result of one frame-to-frame segmentation as initialization for others, realtime performance can be reached.

5 Conclusion

We presented a bottom-up approach for motion segmentation using correspondences from multiple scales. On a frame-to-frame level, they are segmented into different motions using our fast and precise motion-split-and-merge (MSAM) approach. These frame-to-frame correspondences of multiple frames and scales are merged to trajectories. The motion of such trajectories with various lengths, time spans and missing data is segmented by multi-scale motion clustering (MSMC), a single spectral clustering step with Jaccard distances of preferred trajectory motion parameter sets as input. Last, an accurate estimator of the number of motions was presented.

Using complete trajectories from the Hopkins data set as input, the overall error rate was better than any other current state-of-the-art motion segmentation approach which is able to handle missing data without constraints. Furthermore, we estimated the number of motion classes with best precision reported so far. We showed that our approach allows more than 98% data of the input trajectories missing. In a challenging real-world example with low-quality image data and multi-object occlusions, we demonstrated that our algorithm allows fast and accurate motion segmentation of SIFT correspondences on multiple scales.

Acknowledgement. This work was funded by the German Federal Ministry of Education and Research, BMBF project ASEV.

References

1. Lezama, J., Alahari, K., Sivic, J., Laptev, I.: Track to the future: Spatio-temporal video segmentation with long-range motion cues. In: CVPR, pp. 3369–3376 (2011)
2. Ochs, P., Brox, T.: Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In: Proc. ICCV (2011)
3. Sheikh, Y., Javed, O., Kanade, T.: Background subtraction for freely moving cameras. In: Proc. ICCV, pp. 1219–1225 (2009)
4. Brox, T., Malik, J.: Object Segmentation by Long Term Analysis of Point Trajectories. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 282–295. Springer, Heidelberg (2010)
5. Cifuentes, C.G., Sturzel, M., Jurie, F., Brostow, G.J.: Motion models that only work sometimes. In: Proc. BMVC (2012)
6. Sand, P., Teller, S.: Particle video: Long-range motion estimation using point trajectories. IJCV 80, 72–91 (2008)
7. Cheriyadat, A., Radke, R.: Non-negative matrix factorization of partial track data for motion segmentation. In: Proc. ICCV, pp. 865–872 (October 2009)
8. Sivic, J., Schaffalitzky, F., Zisserman, A.: Object level grouping for video shots. IJCV 67(2), 189–210 (2006)
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV 60, 91–110 (2004)
10. Favaro, P., Vidal, R., Ravichandran, A.: A closed form solution to robust subspace estimation and clustering. In: Proc. CVPR, pp. 1801–1807 (2011)
11. Chen, G., Lerman, G.: Motion segmentation by scc on the hopkins 155 database. In: Proc. ICCV Workshop on Dynamical Vision (2009)
12. Yu, J., Chin, T.J., Suter, D.: A global optimization approach to robust multi-model fitting. In: Proc. CVPR (2011)
13. Vidal, R., Hartley, R.: Motion segmentation with missing data using powerfactorization and gpca. In: Proc. CVPR, pp. 310–316 (2004)
14. Rao, S., Tron, R., Vidal, R., Ma, Y.: Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. TPAMI 32, 1832–1845 (2010)
15. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Proc. NIPS, pp. 849–856 (2001)
16. Fradet, M., Robert, P., Perez, P.: Clustering point trajectories with various lifespans. In: Proc. CVMP, pp. 7–14 (2009)
17. Toldo, R., Fusiello, A.: Robust Multiple Structures Estimation with J-Linkage. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 537–547. Springer, Heidelberg (2008)
18. Elhamifar, E., Vidal, R.: Sparse subspace clustering. In: Proc. CVPR, pp. 2790–2797 (2009)
19. Fouhey, D.F., Scharstein, D., Briggs, A.J.: Multiple plane detection in image pairs using j-linkage. In: Proc. ICPR (2010)
20. Jain, R., Kasturi, R., Schunck, B.G.: Machine Vision. McGraw-Hill (1995)
21. Fischler, M.A., Bolles, R.C.: Random sample consensus. CACM 24, 381–395 (1981)
22. Tron, R., Vidal, R.: A benchmark for the comparison of 3-d motion segmentation algorithms. In: Proc. CVPR (2007)
23. Chin, T.J., Wang, H., Suter, D.: The ordered residual kernel for robust motion subspace clustering. In: Proc. NIPS, pp. 333–341 (2009)
24. Chin, T.J., Suter, D., Wang, H.: Multi-structure model selection via kernel optimisation. In: Proc. CVPR, pp. 3586–3593 (2010)
25. Wu, C.: SiftGPU: A GPU implementation of scale invariant feature transform (SIFT) (2007), `http://cs.unc.edu/~ccwu/siftgpu`