CVPR 2012

Higher Order Motion Models and Spectral Clustering

Peter Ochs and Thomas Brox
Computer Vision Group
University of Freiburg, Germany
{ochs, brox}@informatik.uni-freiburg.de

Abstract

Motion segmentation based on point trajectories can integrate information of a whole video shot to detect and separate moving objects. Commonly, similarities are defined between pairs of trajectories. However, pairwise similarities restrict the motion model to translations. Non-translational motion, such as rotation or scaling, is penalized in such an approach. We propose to define similarities on higher order tuples rather than pairs, which leads to hypergraphs. To apply spectral clustering, the hypergraph is transferred to an ordinary graph, an operation that can be interpreted as a projection. We propose a specific non-linear projection via a regularized maximum operator, and show that it yields significant improvements both compared to pairwise similarities and alternative hypergraph projections.

1. Introduction

The need for segmented training data in visual learning is enormous. Even though crowd sourcing provides a cost efficient solution to manually annotate larger datasets, recent trends towards learning subcategories and attributes increase the demand for more and more annotated data – ideally obtained via unsupervised or very weakly supervised object segmentation.

Motion is key in unsupervised object segmentation, as indicated in psychophysical studies with infants and blind persons who gained eyesight as adults [17]. Recently, [4] presented a promising approach based on dense point trajectories. Due to the long term analysis of motion, this approach can reliably extract moving objects in a video, even if the motion in some frames is ambiguous. This adds significantly to classical motion segmentation methods based on two-frame optical flow.

[4] is based on direct, pairwise comparison of trajectories. The pairwise similarities between all trajectories are expressed by an affinity matrix, which can be analyzed by spectral clustering [19, 15]. However, the pairwise analysis comes with a limitation: comparing two trajectories allows to decide only if they belong to the same translational model or not. All motion that deviates from a translational model, for instance rotation or scaling, leads to a non-zero distance. This problem is alleviated by the rather dense set of trajectories used in [4], but it does not go away. Consequently, the method works well if objects move coarsely according to a translational model, but fails in case of dominant complex motion. Multi-body factorization methods, such as [6, 21, 18, 8], can incorporate (affine) higher order motion models easily, but they require all trajectories to be roughly the same length. As motion across many frames leads to significant occlusion/disocclusion phenomena, the factorization framework is restricted to much shorter shots.

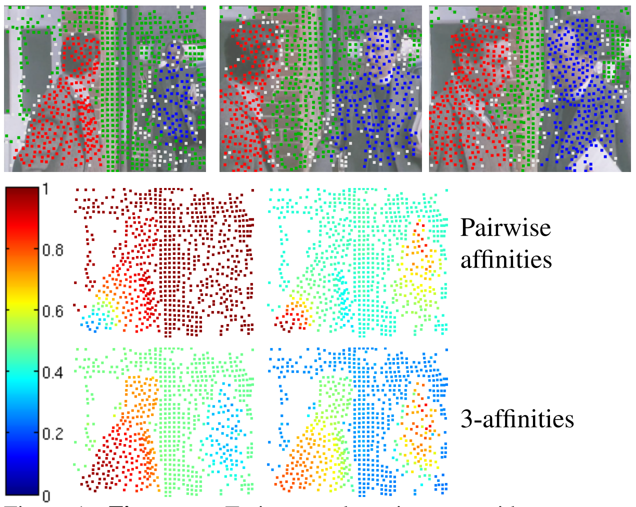


Figure 1. **First row:** Trajectory clustering on a video sequence with 800 frames. The approaching man labeled in blue generates a scaling motion that cannot be described properly with a translational model and pairwise affinities. Our spectral clustering model based on triplets describes similarity transformations and can capture it easily. **Bottom rows:** The top eigenvectors of the pairwise clustering model are far from being piecewise constant, whereas thresholding the topmost eigenvector of the higher order model is already sufficient in this example.

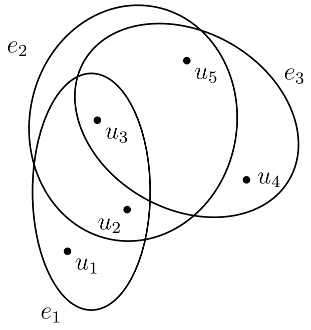
In this paper, we are aiming for the best of both models: we keep the flexible comparison of trajectories of various length, and we allow for higher order motion models that do not penalize rotation and scaling anymore. Our approach can work with motion models of arbitrary order, though in our experiments we have focused on in-plane rotation and scaling, i.e., 2D similarity transformations without reflections. To uniquely determine the parameters of such a model, two motion vectors are required. Obviously pairwise affinities are not applicable: a pair of motion vectors always fits a similarity transformation perfectly. A third motion vector is necessary to decide on the compatibility of a triplet. Rather than pairwise affinities, we obtain tertiary affinities. The affinity matrix becomes an affinity tensor and the underlying graph a hypergraph.

To cluster trajectories based on a higher order motion model we need a hypergraph version of spectral clustering; but what is the normalized Laplacian of a hypergraph? All definitions of hypergraph Laplacians come down to a projection of the hypergraph to an ordinary graph, on which the classical Laplacian is trivially applied [1]. The projections applied in practice end up in setting the edge weights to the sum over all corresponding hyperedge weights. We argue that at least in the case of motion segmentation this is not a good projection, because many triplets in the hyperedge set will comprise multiple objects though the considered pair just covers a single object. This can be particularly problematic in case of small objects. For this reason, we propose a regularized maximum projection, which just requires one of the triplets to cover the same object. Since hypergraphs come with a larger computational complexity than ordinary graphs, we also present a subsampling strategy that leads to acceptable computation times while not losing the effect of the higher order model.

Once the affinities for the ordinary graph are computed, we apply spectral clustering with spatial regularity as proposed in [4]. The clustering comes with an automatic selection of the model parameters, i.e., we do not need to specify the number of objects. Our experiments show clearly improved results on the Berkeley motion segmentation benchmark with regard to the translational model but also with regard to a projection based on the sum of hyperedges.

2. Related Work

Opposed to VLSI design, where hypergraph partitioning has been used for decades [3], hypergraphs have appeared in computer vision only recently. One of the first papers is the one by Agarwal et al. [2], who analyzed a set of existing hypergraph partitioning methods and showed their application to illumination invariant clustering of faces. In [1] they showed that all existing methods for spectral clustering on k-uniform hypergraphs can be expressed as clique averaging, which means that the hypergraph is projected onto an ordinary graph and spectral clustering is applied there.



$V \setminus \mathcal{E}$	e_1	e_2	e_3
u_1	1	0	0
u_2	1	1	0
u_3	1	1	1
u_4	0	0	1
u_5	0	1	1

Figure 2. Example of a hypergraph (left) and its incidence relationships h (right), e.g., $h(u_1, e_1) = 1$ and $h(u_4, e_2) = 0$. A hypergraph describes the relation among an arbitrary number of vertices, here triples.

Their proof includes the method in [22], and it is easy to see that [10, 5] are also approximations of this formulation based on sampling. The general p-norm introduced in [2] includes our maximum projection for $p \to \infty$. These previous works are mainly concerned with a general hypergraph formulation and consider applications as a proof of concept. In contrast, the present paper comes with a specific application that requires hypergraphs, and we argue why the maximum projection fits this application much better than the common average projection.

Hypergraphs have become popular also in conjunction with inference in higher order MRFs [12, 11]. Like here, the goal of using hypergraphs is to consider larger cliques. However, due to the Markov property, vertices are only locally connected by hyperedges, which is in contrast to this paper, where hyperedges cover the graph globally.

Although [1] argue that all existing spectral clustering on hypergraphs can be expressed on ordinary graphs, a hypergraph partitioning method based on tensor algebra that cannot be reduced to ordinary graph partitioning is conceivable. Apart from generalized power iteration for graph matching [7], there are many theoretical works in the tensor algebra literature [9, 14, 13].

3. Hypergraph Modelling

A *hypergraph* $\mathcal{H} = (V, \mathcal{E})$ consists of a *vertex* set V and a *hyperedge* set $\mathcal{E}$ of subsets of V. The number of vertices and hyperedges is finite and given by the *cardinality* $|\cdot|$ of the respective set. A vertex $v \in V$ and a hyperedge $e \in \mathcal{E}$ are called *incident* if $v \in e$. We represent this relationship by an indicator function $h\colon V \times \mathcal{E} \to \{0, 1\}$. Figure 2 shows an example.

We focus on *undirected*, *weighted*, k-*uniform* hypergraphs, i.e., the ordering of vertices in a hyperedge does not matter, each hyperedge e is assigned a weight $w_{\mathcal{H}}(e) \in \mathbb{R}_0^+$ and the number of vertices in a hyperedge (the *degree*) $\delta_{\mathcal{H}}(e) := |e| = \sum_{v \in V} h(v, e) = k$ is constant. In the

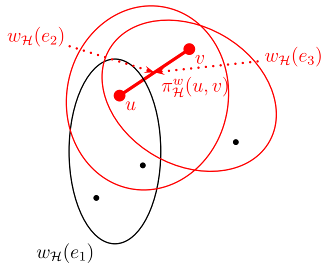


Figure 3. A hypergraph which shows the hyperedge weights (ellipses) that are projected to the edge (line) between the vertices u and v. The hyperedges incident with both vertices contribute to the weight of the projected edge.

special case $k = 2$, hypergraphs become ordinary graphs. As the weights of 2-hypergraphs are represented by an $\mathbb{R}^{|V|^2} := \mathbb{R}^{|V|\times|V|}$ affinity matrix, weights of k-uniform hypergraphs are represented by a $\mathbb{R}^{|V|^k} := \mathbb{R}^{|V|\times\dots\times|V|}$ affinity tensor. For undirected hypergraphs the affinity tensor is a symmetric k-order tensor with non-negative entries, where *symmetry* means that the entries of all index permutations are the same.

3.1. Projecting the Hypergraph to its Primal Graph

To partition a hypergraph via spectral clustering, we must project it to an ordinary graph. Therefore, we define the *primal graph* that consists of the same vertex set as the hypergraph but its edge set connects each pair of vertices incident with the same hyperedge.

Define the *projection operator* $\pi^w_{\mathcal{H}} \colon V \times V \to \mathbb{R}^+_0$, which assigns to each pair of vertices a weight by projecting the weights of all hyperedges incident with both of the vertices; see Figure 3. Appropriate projection operators are positive, symmetric, and monotone [2].

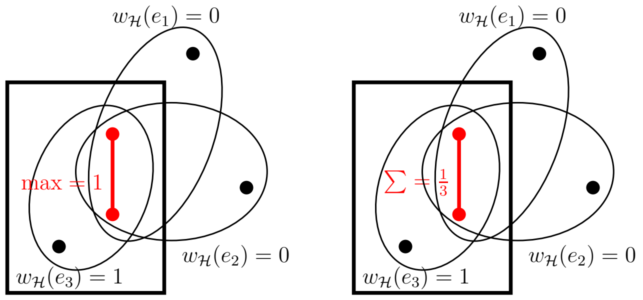
Based on such a projection operator, we can write the projection from an affinity tensor to an affinity matrix as

$$\begin{aligned} \Pi_{\mathcal{H}} \colon \mathbb{R}^{|V|^k} &\to \mathbb{R}^{|V|\times|V|}, \\ w_{\mathcal{H}} &\mapsto \Pi_{\mathcal{H}}(w_{\mathcal{H}}) := (\pi^w_{\mathcal{H}}(u,v))_{u,v\in V}. \end{aligned} \tag{1}$$

The common projection is via summation:

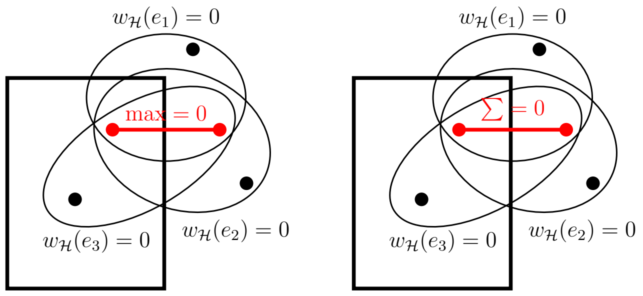
$$\pi^w_{\mathcal{H}}(u,v) = \sum_{e\in\mathcal{E}} \frac{w_{\mathcal{H}}(e)}{\delta_{\mathcal{H}}(e)} h(u,e)h(v,e). \tag{2}$$

It has been proposed, for instance, in [2, 22]. Let $H := (h(u,e))_{u\in V, e\in\mathcal{E}} \in \mathbb{R}^{|V|\times|\mathcal{E}|}$ be the incidence matrix and $D_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ the hyperedge degree diagonal matrix. Then (2) can be written as $HWD_{\mathcal{E}}^{-1}H^\top$, which is the affinity matrix in [22]. This means, the hypergraph Laplacian in [22] fits in this projection framework as a special case [1].



$w_{\mathcal{H}}(e_1) = 0$
$\max = 1$
$w_{\mathcal{H}}(e_2) = 0$
$w_{\mathcal{H}}(e_3) = 1$
$w_{\mathcal{H}}(e_1) = 0$
$\sum = \frac{1}{3}$
$w_{\mathcal{H}}(e_2) = 0$
$w_{\mathcal{H}}(e_3) = 1$

Figure 4. max-projection vs. $\sum$-projection if the edge covers a single object. One additional trajectory on the object indicates that the motion model is consistent. The $\sum$-projection yields the average affinity of all hyperedges, which is way too low.



$w_{\mathcal{H}}(e_1) = 0$
$\max = 0$
$w_{\mathcal{H}}(e_3) = 0$
$w_{\mathcal{H}}(e_2) = 0$
$w_{\mathcal{H}}(e_1) = 0$
$\sum = 0$
$w_{\mathcal{H}}(e_3) = 0$
$w_{\mathcal{H}}(e_2) = 0$

Figure 5. max-projection vs. $\sum$-projection if the edge covers two different objects. Both projections yield the optimal affinity.

3.2. max-Affinity Projections for Motion Segmentation

In [2] a class of functions parameterized by $p \in \mathbb{R}^+$ is proposed. For $p = 1$ this corresponds again to the $\sum$-projection in (2). In the following, we will argue that a larger p, in particular $p \to \infty$ that corresponds to the maximum operator, is more appropriate for motion segmentation.

In motion segmentation, the hypergraph's vertices are point trajectories and the hyperedges are k-tuples of these trajectories. For each k-tupel of trajectories we define an affinity based on their motion similarity; details will follow in Section 4. According to the definition in (1), the max-projection reads

$$\pi^w_{\mathcal{H}}(u,v) = \underset{\substack{w_{\mathcal{H}}(e) \\ u,v\in e\in\mathcal{E}}}{\operatorname{argmax}} \, w_{\mathcal{H}}(e)\, l(e), \tag{3}$$

where we include a weight $l(e)$ that is the number of common frames of all trajectories in e. This weighting treats longer, strongly overlapping trajectories as more reliable.

In Figure 4, 5 let us analyze two important cases in the motion segmentation scenario: (1) both vertices of a pairwise edge lie within a single object; (2) they cover two different objects. For simplicity, we focus on hyperedges of degree 3. The first case reveals the advantage of projecting with the $\max$-operator: a single third vertex in the object is sufficient for a high affinity, whereas the $\sum$-projection leads to a bad compromise.

In the second case both vertices of the pairwise edge belong to different objects and we want the affinity to be low. The motion model suggested by the two vertices is not compatible with one of two object motions and so there is no hyperedge with a high affinity for this pair. Both projections lead to the correct affinity.

These considerations are only valid in the noise-free case and if the motion subspaces do not overlap. In contrast to the sum, the maximum operator is unstable. A single outlier or a non-empty cut of the two motion subspaces will spoil the result in case (2). Both problems exist in practice. For this reason, we must regularize the $\max$-operator by adding the following condition: if the projection yields a high affinity, but 90% of the considered hyperedges have 0 affinity, we assign 0 affinity to the pairwise edge. This condition makes the use of the $\max$-operator stable enough for our motion segmentation task. Other types of regularization are conceivable as well, e.g., an L_p-norm with a finite, sufficiently large p.

4. Computing Hyperedge Affinities

From the tracker in [20] we obtain n trajectories c_i, $i = 1, \ldots, n$ in a video with M frames. Most trajectories do not cover all M frames due to occlusion/disocclusion. If a trajectory c_i exists at a frame $t \in \{1, \ldots, M\}$, it comes with pixel coordinates $(x(t), y(t))^\top$ at this frame. Consider a k-tuple e of trajectories $c_{i_1}, \ldots, c_{i_k}$. For each such tuple we compute a distance $d\colon \mathcal{E} \to \mathbb{R}_0^+$, which is converted into an affinity via

$$w_{\mathcal{H}}(e) := \exp\left(-\lambda d(e)\right) \tag{4}$$

with $\lambda = 0.1$.

4.1. Computing 3-Distances

In our experiments we focus on 3-tuples. Like in [4], we compose the distance d for each triplet (c_i, c_j, c_k), $i, j, k \in \{1, \ldots, n\}$ of the distances $d^{(t)}$ in all *common frames* t, i.e., frames in which all three trajectories are visible.

For a fixed $t \in \{1, \ldots, M\}$ we estimate the error of the underlying motion model according to the change from frame t to $t' := t + 8$. If less than 8 common frames are available we restrict the computation to the common frames.

The incentive of considering three trajectories at a time is to have Euclidean translations, rotations, and scalings without penalty. Formally, these movements can be described by the group of special similarity transformations $\mathrm{SSim}(2)$. In contrast to the group of similarity transformations $\mathrm{Sim}(2)$ it excludes reflections.

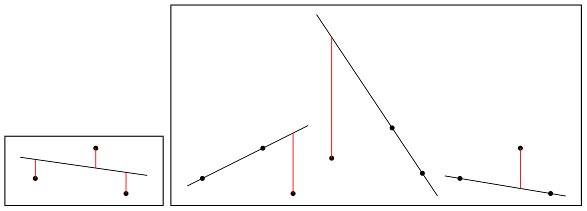


Figure 6. **Left: (a)** Fitting a linear model to three points via least squares. Even though the points do not fit a line, the total error (in red) is small. **Right: (b)** A line is fit to all pairs of points and the error is measured based on how well the third point fits the line. The maximum discrepancy among all tuples clearly indicates that the points do not fit a common model.

Let c_i, c_j be two of three trajectories. The motion model $\mathcal{T}_{i,j}(t) \in \mathrm{SSim}(2)$ is described by a scaling parameter s, a rotation matrix R_α, and the translation vector $\mathbf{v} := (v_1, v_2)^\top$. These parameters can be computed uniquely from the coordinates at frames t and t' as follows:

$$\begin{aligned} s &= \tfrac{\|c_i(t')-c_j(t')\|}{\|c_i(t)-c_j(t)\|} \\ \alpha &= \arccos\left(\tfrac{\left(c_i(t')-c_j(t')\right)^\top (c_i(t)-c_j(t))}{\|c_i(t')-c_j(t')\|\|c_i(t)-c_j(t)\|}\right) \\ \mathbf{v} &= \tfrac{1}{2}\left(\left(c_i(t') + c_j(t')\right) - sR_\alpha\left(c_i(t) + c_j(t)\right)\right). \end{aligned} \tag{5}$$

We can test how well the third trajectory c_k fits this transformation by the L_2-distance $\|\mathcal{T}_{i,j}c_k(t) - c_k(t')\|$.

Obviously, the motion model could be computed also from c_i, c_k or from c_j, c_k. As illustrated in Figure 6b the choice of the pair has a large impact on the distance. One could also consider estimating the optimum model from all three trajectories in the least-squares sense, as shown in Figure 6a. However, this reduces the distance of an incompatible triplet such that it cannot be distinguished from noise in a compatible one anymore. For this reason, we pick the maximum distance among all 2-tuples $(u, v) \in \{(i,j), (i,k), (j,k)\}$ with the third trajectory c_w, $w \in \{i,j,k\} \setminus \{u,v\}$:

$$d^{(t)}(i,j,k) := \max d^{(t)}_{\text{ratio}}(u,v)\,\|\mathcal{T}_{u,v}c_w(t) - c_w(t')\|. \tag{6}$$

If the three trajectories do *not* fit the same model, it will be very large leading to a clear separation of motion clusters. The weight

$$d^{(t)}_{\text{ratio}}(i,j) := \left(\tfrac{1}{2}\left(\tfrac{\|c_i(t)-c_j(t)\|}{\|c_i(t)-c_k(t)\|} + \tfrac{\|c_i(t)-c_j(t)\|}{\|c_j(t)-c_k(t)\|}\right)\right)^{\frac{1}{4}}. \tag{7}$$

is introduced to avoid numerical problems. When a motion model $\mathcal{T}_{i,j}(t)$ is estimated based on very nearby trajectories, small numerical errors can cause large errors at the location of distant trajectories.

Like in [4] we normalize distances with the optical flow variance in the image and weight them by the spatial distance of trajectories. For the final distance of the 3-tuple we take the maximum distance over all common frames $d(i,j,k) = \max_t d^{(t)}(i,j,k)$.

4.2. Higher Order Affinities

The above framework extends easily to more general motion models. For instance, distances based on an affine motion model could be computed from 4-tuples of trajectories. While this fits 3D rigid motions even better, it also further increases the computational costs. With 3-affinities we have cubic complexity $\mathcal{O}(n^3)$. The complexity of a k-tuple is $\mathcal{O}(n^k k)$, where k is due to the maximum in (6). In the next section, we present a sampling strategy that reduces the complexity of 3-affinities to $\mathcal{O}(n^2)$ without significant degradation compared to the full model. While k-affinities with $k > 3$ are an interesting future option, they also come with other issues. For example, as 3D rotations lead to self-occlusion, trajectories are usually too short for a more complex model to show advantages. This is why we consider only 3-affinities in the remainder of this paper.

5. Sampling Strategy

We propose a combination of deterministic and random sampling to reduce the number of hyperedges to be considered, which makes the approach tractable in case of large numbers of trajectories. For each pairwise edge, we sample hyperedges comprising both vertices of the edge and one additional vertex. For both vertices in each pairwise edge, we take the 12 spatially nearest neighbours and, additionally, 30 vertices randomly as third vertex. It is worth noting that we sample only over the third vertex, whereas the graph is spanned globally over all trajectories, which is in contrast to MRF approaches. The mixture of deterministic and random sampling ensures finding enough relevant triplets.

We verified this in a synthetic experiment shown in Figure 7. The results for the full hypergraph, where all possible triplets are considered, and two sampling strategies are reported in Table 1.

The evaluation shows that sampling degrades the accuracy only a little when some objects are much smaller than others. In some special cases, sampling is even advantageous. If the smaller objects cover exactly 10% of all trajectories, 90% of all triplets comprise a vertex outside the object and, thus, yield 0 affinity. Since our regularized max-projection treats this situation erroneously as outlier, it assigns 0 affinity to the pairwise edge in case of the full graph. For the random sampling, it is sufficient to have 2 ($> \frac{12}{10}$) triplets covering the object. Since only a small subset of the pairwise edges needs high affinities, sampling such a subset is very likely. Of course this probability decreases as the object shrinks. In order to improve the performance in object segmentation (cf. Table 2), where certain compactness assumptions hold, we complement the random sampling by a deterministic nearest neighbor sampling.

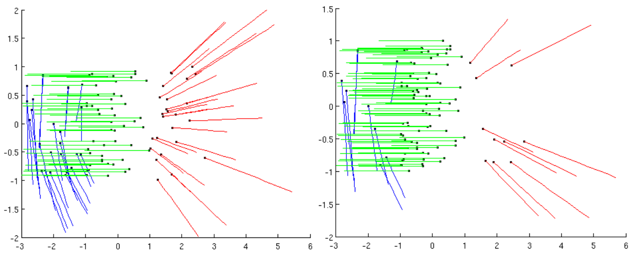


Figure 7. Two synthetic experiments with 100 trajectories and 3 regions of different motion and size (rotation, translation, scaling). In the experiment we reduced the size of the left and right region successively to 25, 20, ..., 5, 4, and 3 trajectories simulating the effect of small objects and a large background.

	25	20	15	10	5	4	3
full	100	100	100	84	96	100	99
rnd	100	100	100	99.6	97.8	94.8	96
rnd+nn	100	100	100	100	97.2	97.6	96.3

Table 1. Results of the experiment in Figure 7. The number of trajectories in the left and the right cluster is given in the first row. The table shows the percentage of correctly clustered trajectories for the full graph (full), random sampling (rnd) with 12 samples, and a combination (rnd+nn) of 4 random samples and 4 nearest neighbors at a time. The results on rnd and rnd+nn are averaged over 10 evaluations. Sampling does not affect the accuracy much, even for small objects, but reduces the complexity considerably.

6. Evaluation

6.1. Berkeley Motion Segmentation Benchmark

We compare to multi-body factorization [18] and the translational motion model from [4] on the benchmark dataset introduced in [4]. We used the evaluation code provided with the benchmark. Results are shown in Table 2. The *density* is the percentage of labeled pixels. The *overall clustering error* measures the percentage of erroneous labels per labeled pixel, and the *average clustering error* the pixel error per region averaged over all regions. The evaluation also reports an *over-segmentation* penalty, which is the number of merging events per ground truth region necessary to make the labeling fit the ground truth annotation.

We use the same tracker as in [4], which allows to adjust the density. We run the tracker with 4- and 8-spacing, meaning that, the tracker samples only every 4th or 8th pixel, respectively, in each direction.

Performance differences become clearly visible when considering all frames. As already shown by [4], multi-body factorization cannot deal with large occlusions, and this shows in the numbers. Comparing the pairwise affinities [4] to our higher order model reveals a significant improvement of 50% in the overall error. The higher quality also shows in more objects being extracted correctly (de-

	Density	overall error	average error	over-segmentation	extracted objects
All available frames					
ALC corrupted [18]	0.99%	5.32%	52.76%	0.10	15
ALC incomplete* [18]	3.29%	14.93%	43.14%	18.80	5
pairwise affinities [4]	3.31%	6.52%	27.29%	2.12	29
pairwise affinities* [4]	3.28%	6.59%	26.69%	1.44	27
$\sum$-projection* (rnd+nn)	3.22%	5.36%	20.08%	2.08	31
max-projection* (rnd+nn)	**3.22%**	**4.48%**	**22.34%**	**1.84**	**31**
pairwise affinities [4]	0.79%	6.78%	25.48%	1.73	30
$\sum$-projection (rnd+nn)	0.78%	6.12%	22.71%	2.31	30
$\max$-projection (rnd)	0.77%	7.32%	32.95%	1.92	22
max-projection (rnd+nn)	**0.78%**	**4.33%**	**21.96%**	**1.58**	**34**

Table 2. Evaluation on the Berkeley motion segmentation benchmark. Entries marked with "*" are evaluated without the sequence *marple7*. The upper part of the table reports on a 4-spacing, the lower part on an 8-spacing. (rnd) refers to a purely randomized sampling using 54 samples. (rnd+nn) uses 30 random samples and for each vertex the 12 spatial nearest neighbors.

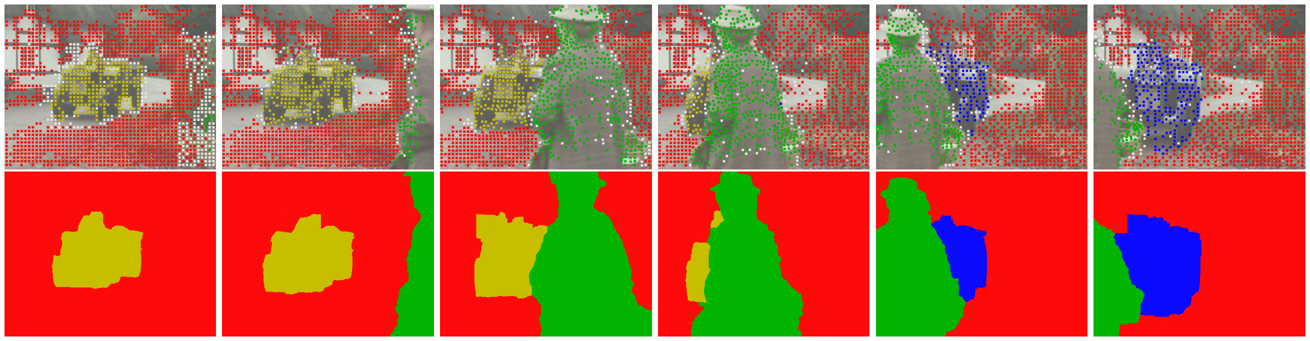


Figure 8. **Top row:** Result on *marple8* obtained with the proposed method and a spatial subsampling of 8. **Bottom row:** Dense interpolation of this result using the binaries from [16].

fined by [4] as regions with less than 10% error minus the background region). Figures 9 and 10 show a qualitative comparison.

We also compared the proposed max-projection to the $\sum$-projection that is used by previous works on hypergraph partitioning, e.g., in [2]. The $\sum$-projection performs better than just the pairwise affinities, but only the max-projection can fully exploit the higher order affinities. In fact, most of the increase in performance is not due to the use of hypergraphs as such, but due to hypergraphs in conjunction with the right projection method. Figure 11 shows an advantage of our $\max$-projection over the $\sum$-projection with respect to the leakage problem explained in Figure 4.

Figure 8 shows our result for marple8 and a dense segmentation obtained using the code from [16].

Thanks to the sampling, the computational complexity is still in $\mathcal{O}(n^2)$, yet practical computation times increase over the translational model. While clustering the whole sequence of *cars1* with 4850 trajectories runs in 50s on a single core with pairwise affinities, 3-affinities require 48 minutes. The computation of affinities can be perfectly parallelized on the GPU though. With an expected speedup of about 50× this would result in approximately 3s per frame, which is well tractable also in a large scale task.

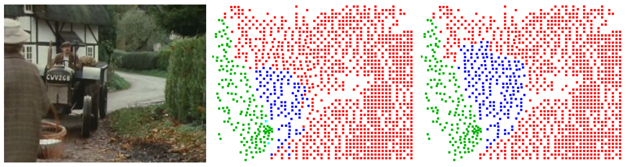


Figure 9. **From left to right**: frame 46 of marple8, result with pairwise affinities, our result.

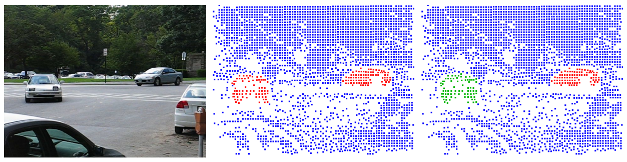


Figure 10. **From left to right**: frame 21 of cars5, result with pairwise affinities, our result.

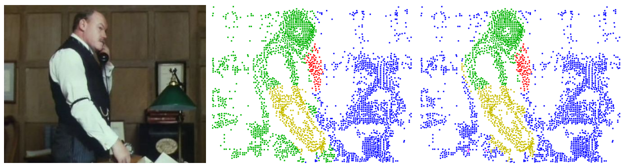


Figure 11. **From left to right**: frame 65 of marple13, result with $\sum$-projection, our result (max-projection).

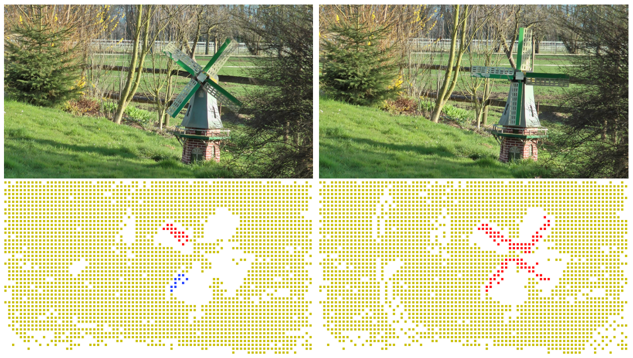


Figure 12. **Upper row:** Windmill with rotating sails. **Bottom left:** Result with pairwise affinities [4]. **Bottom right:** Our result with 3-affinities.

6.2. Other Real World Examples

The Berkeley motion segmentation benchmark focuses mainly on translational motion. Hence, we collected some additional videos with more complex motion and compare our method to [4].

The advantage of higher order motion models becomes very clear in Figure 12. It is not possible to describe the rotation of the windmill sails with pairwise affinities. Consequently, trajectories are either treated as outliers and are removed completely, or they lead to over-segmentation. The higher order model can handle this example easily. The same effect can be seen in Figure 13. Although the 3D rotation of the monkey is not without penalty in our 3-affinity model, the penalty with pairwise affinities is even larger. Figure 14 shows a very difficult sequence with camera zoom. Some ducks move in a very similar manner and there is strong articulation. Pairwise affinities lead to a single cluster. In contrast, 3-affinities can capture the moving ducks quite well. Executables are provided to allow evaluations on other sequences.

7. Conclusions

We have presented a higher order motion segmentation method based on spectral clustering on hypergraphs. We showed that this model can handle the limitations of pairwise affinities very well. As a very interesting observation in this work we note that the use of hypergraphs alone is not sufficient. The usual $\sum$-projection hardly improved results over pairwise affinities. It is actually the choice of the projection from the hypergraph to the ordinary graph that makes the difference.

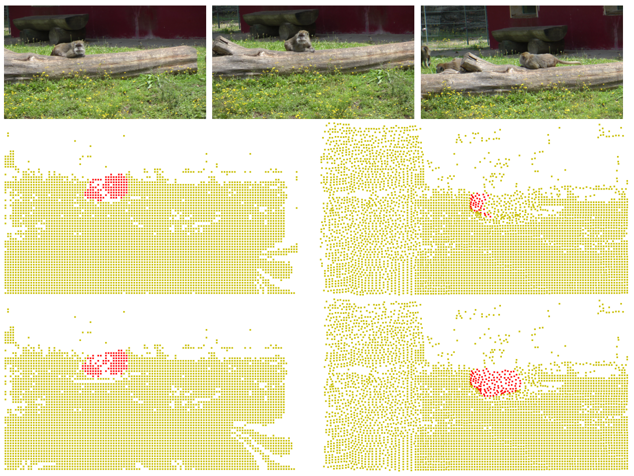


Figure 13. **First row:** Monkey that lies in the beginning, stands up, and turns to the other side. **Second row:** Result with pairwise affinities [4] for the first and last frame. **Third row:** Our result with 3-affinities. This examples shows that 3-affinities also better fit 3D rotations than pairwise affinities, though not explicitly modeled.

Acknowledgment

We gratefully acknowledge funding by the German Research Foundation (DFG) via grant BR 3815/5-1.

References

[1] S. Agarwal, K. Branson, and S. Belongie. Higher order learning with graphs. In *International Conference on Machine Learning (ICML)*, 2006. 2, 3

[2] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 2, 3, 6

[3] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: a survey. *Integration: The VLSI Journal*, 1995. 2

[4] T. Brox and J. Malik. Object segmentation by long-term analysis of point trajectories. In *European Conference on Computer Vision (ECCV)*, 2010. 1, 2, 4, 5, 6, 7

[5] G. Chen and G. Lerman. Spectral curvature clustering SCC. *International Journal of Computer Vision*, 2009. 2

[6] J. Costeira and T. Kanande. A multi-body factorization method for motion analysis. In *International Conference on Computer Vision (ICCV)*, 1995. 1

[7] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011. 2

[8] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 1

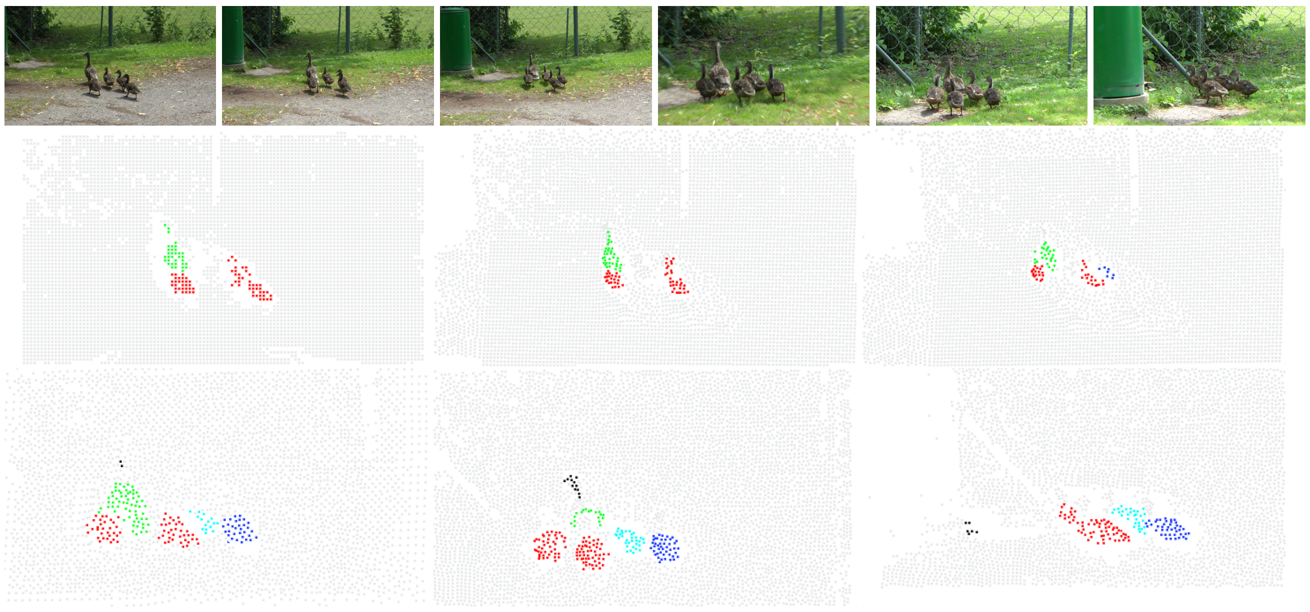


Figure 14. **Top row:** A family of ducks passing the camera. While they are moving away, the camera starts zooming on them. Finally the mother vanishes behind a green box. Pairwise affinities cannot model the zooming motion of the camera properly. This results in a single cluster comprising the ducks and the background. **Other rows:** Our 3-affinities are invariant to scaling. The ducks can be separated from the background. The method even separates most of the single ducks.

[9] E. Gnang, A. Elgammal, and V. Retakh. A spectral theory for tensors. *ArXiv e-prints*, 2010. 2

[10] V. Govindu. A tensor decomposition for geometric grouping and segmentation. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 2

[11] H. Ishikawa. Higher-order clique reduction in binary graph cut. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[12] P. Kohli, M. P. Kumar, and P. Torr. P3 & beyond: solving energies with higher-order cliques. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 2

[13] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Journal on Applied Mathematics*, 2009. 2

[14] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Applied Mathematics*, 2000. 2

[15] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, 2002. 1

[16] P. Ochs and T. Brox. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In *International Conference on Computer Vision (ICCV)*, 2011. 6

[17] Y. Ostrovsky, E. Meyers, S. Ganesh, U. Mathur, and P. Sinha. Visual parsing after recovery from blindness. *Psychological Science*, 2009. 1

[18] S. R. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 1, 5, 6

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000. 1

[20] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *European Conference on Computer Vision (ECCV)*, 2010. 4

[21] J. Yan and M. Pollefeys. A general framework for motion segmentation: independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conference on Computer Vision (ECCV)*, 2006. 1

[22] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: clustering, classification, and embedding. In *Advances in Neural Information Processing Systems (NIPS)*, 2007. 2, 3