Clustering Point Trajectories with Various Life-Spans

Matthieu Fradet, Philippe Robert
Thomson R&D France SNC
1, av. de Belle Fontaine - CS 17616
35576 Cesson Sévigné - France
{matthieu.fradet, philippe.robert}@thomson.net

Patrick Pérez
INRIA Rennes-Bretagne Atlantique
Campus universitaire de Beaulieu
35042 Rennes Cedex - France
Patrick.Perez@inria.fr

Abstract—Motion-based segmentation of a sequence of images is an essential step for many applications of video analysis, including action recognition and surveillance. This paper introduces a new approach to motion segmentation operating on point trajectories. Each of these trajectories has its own start and end instants, hence its own life-span, depending on the pose and appearance changes of the object it belongs to. A set of such trajectories is obtained by tracking sparse interest points. Based on an adaptation of recently proposed J-linkage method, these trajectories are then clustered using series of affine motion models estimated between consecutive instants, and an appropriate residual that can handle trajectories with various life-spans. Our approach does not require any completion of trajectories whose life-span is shorter than the sequence of interest. We evaluate the performance of the single cue of motion, without considering spatial prior and appearance. Using a standard test set, we validate our new algorithm and compare it to existing ones. Experimental results on a variety of challenging real sequences demonstrate the potential of our approach.

Keywords- motion segmentation; point trajectories; clustering; J-linkage

I. INTRODUCTION

Segmenting a sequence of images in regions of similar motion is a very important processing step required by different applications such as video compression, interactive video object manipulation and surveillance. The resolution of the underlying problem is really fundamental for dynamic scene understanding. The importance and the variety of the possible applications make the problem be an active topic in computer vision.

Sparse methods cluster a sparse set of point trajectories. They are convenient for applications that do not require any accurate extraction of object boundaries (e.g. object detection, action recognition, tracking or surveillance). Sparse methods often estimate the trajectories along the whole sequence before clustering the obtained set of trajectories.

Under the affine camera model assumption, motion segmentation can be achieved by clustering point trajectories into different linear motion subspaces of low dimension [9, 12, 14]. In these methods, the number of clusters is assumed to be known. A benchmark for the comparison of such algorithms is presented in [11]. Note that for [9, 14] all trajectories are good and complete, i.e. the point tracker never fails during trajectories estimation, "outliers" are tracked features that do not correspond to any particular motion, and all trajectories have the same life-span corresponding to the whole sequence.

However it is very common in feature tracking that a point cannot be tracked during the whole sequence. In particular, occlusions and disappearances from the field of view lead to incomplete trajectories. Pose, illumination or appearance changes can also result in losses of point trackers. As a result, typical point trajectories have variable life-spans. Several approaches have been proposed to handle such sets of trajectories.

In [12], the incomplete data are projected onto a five-dimensional subspace using an adaptation of power factorization.

Another subspace separation algorithm robust to the presence of outlying, incomplete or corrupted trajectories is presented in [7]. Assuming that there are some complete trajectories for each motion cluster, the incomplete trajectories are individually completed prior to subspace separation using "the fewest possible complete trajectories, which will in general be from only one of the subspaces".

A similarity measure based on the Longest Common Subsequence (LCSS) between two trajectories is used in [3] for grouping similar motion trajectories in a hierarchical agglomerative clustering algorithm. Motions that can be handled by this "bottom-up" approach are nevertheless mainly translational.

Extrapolation of incomplete trajectories is performed in [2] for clustering purposes. But, as the aim of this algorithm is the detection of individual entities and not their tracking, clustering is done separately for each frame, which induces temporal inconsistency.

In another spirit, a real-time incremental framework in [6] uses reference frames for creating and maintaining groups of features. Reference frames (one per group) are updated adaptively to the dynamic behavior of the features so that objects can be detected regardless of their speed. However, because images are processed sequentially, segmentation is done regarding the past only, which means that the algorithm may require several frames to estimate the actual number of motion groups.

Dense methods for motion segmentation provide a complete labelling of all pixels in the video. So they are

convenient for applications like video matting, compositing, or object removal. There are many approaches to get dense segmentation maps. Since dense motion-based segmentation is not the focus of our work, only some examples using a sparse approach as initialization step are mentioned below.

In [5], authors introduce the use of normalized correlation between trajectories and of spectral clustering to group the trajectories into a predefined number of clusters.

Although not relying on point trajectories but on point correspondences between two distant images, an original two-step variant of RANSAC is proposed in [13] to estimate multiple motions from the set of these point correspondences. This algorithm performs better than iterations of standard RANSAC [4] with removal of inliers from the data set at each step.

One main contribution of our paper is the introduction of an explicit affine trajectory model and of an appropriate motion-based residual that can handle trajectories with various life-spans. Based on this, we present a new approach to motion-based segmentation using a set of trajectories obtained by tracking sparse interest points over varying time intervals. The clustering is based on the adaptation of the recently proposed J-linkage method [10], which is an agglomerative method where RANSAC is used to generate a large number of initial hypotheses. Another important feature of our system is that it does not require the completion of any trajectories and does not even assume that for each cluster there are points tracked from the beginning to the end of the sequence. We do not consider any spatial prior and do not use appearance since we want to evaluate the efficiency of the single cue of motion.

The paper is organized as follows. Section II introduces the notations and the model. Section III presents our clustering algorithm and a discussion about outliers. Experimental results on a standard test set and on other various sequences are presented in Section IV.

II. Problem Formulation

Let's assume that a total number of P interest points are tracked within a sequence of F frames. Let $\mathbf{f}^{(i)}=[f_t^{(i)}]_{t=s(i)\dots e(i)}$ be the trajectory of the i-th feature point ($i\in\{1\dots P\}$), where $s(i)$ and $e(i)$ are its start and end instants, and $f_t^{(i)}$ stands for the (x,y) coordinates of the point at instant t.

Let k be the number of clusters. For each cluster j, $j=1,\dots,k$, assuming that its motion in the image plane between two consecutive frames can be approximated by an affine motion model, we introduce the affine trajectory model as a time series of forward/backward pairs of 6-parameters affine motion models:

$$M^{(j)}=((F_{s(j)}^{(j)},B_{s(j)}^{(j)}),\dots,(F_{e(j)-1}^{(j)},B_{e(j)-1}^{(j)})), \qquad (1)$$

where $s(j)$ and $e(j)$ are the first and the last instants at which the object j is visible (at least partially), $F_t^{(j)}$ is the forward affine model corresponding to the motion of the cluster j between instants t and $t+1$ and $B_t^{(j)}$ is the backward affine model corresponding to the motion of the cluster j between instants $t+1$ and t. Note that we do not assume any motion constancy or periodicity.

Given the set of trajectories $\mathbf{f}^{(i)}$, $i=1,\dots,P$ of various life-spans, the problem of motion segmentation consists in clustering the dataset into k clusters whose motion is well described by series of affine motion models, and in estimating all the motion parameters of each cluster.

In the general case (since we do not assume any motion constancy or periodicity) it does not make sense to compare motion models from different instants. That is why we consider only the common time section of a trajectory and a model when computing the distance between both. Moreover, since the estimation of an affine model requires at least three pairs of points, we consider only time sections common to at least three trajectories when estimating a model from a set of trajectories (see Fig. 1).

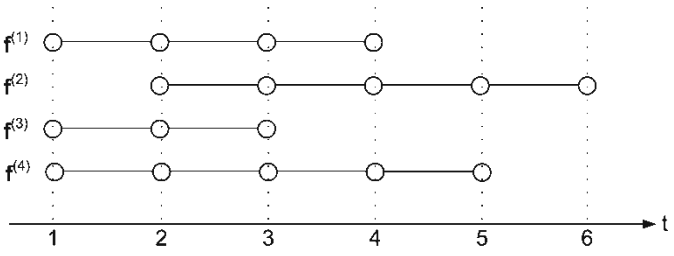

Figure 1. Common time sections of trajectories. Assume that point trajectories $\mathbf{f}^{(1)}$, $\mathbf{f}^{(2)}$, $\mathbf{f}^{(3)}$ belong to the same cluster. We can estimate an affine model between instants 2 and 3 only. If $\mathbf{f}^{(4)}$ joins the cluster, the time series of affine models can now be estimated/updated between instants 1 and 4.

We define the motion residual $r(\mathbf{f},M)$ between a trajectory $\mathbf{f}$ with life-span $\{s(\mathbf{f})\dots e(\mathbf{f})\}$ and a model M with life-span $\{s(M)\dots e(M)\}$ as follows:

$$r(\mathbf{f},M)=\begin{cases}\min_{t\in[s',e']} d(\mathbf{f}',M'(\mathbf{f}',t)) & \text{if } e'-s'>0\\ +\infty & \text{otherwise}\end{cases}, \qquad (2)$$

where $\mathbf{f}'$ is the restriction of $\mathbf{f}$ to its common time section $\{s'\dots e'\}$ with M, and where $s'=\max(s(\mathbf{f}),s(M))$ and $e'=\min(e(\mathbf{f}),e(M))$.

$M'(\mathbf{f}',t)$ is the warp of the trajectory $\mathbf{f}'$ according to the model M' and generated from the coordinates of the reference instant t by cumulating the forward and the backward affine motion models:

$$M'(\mathbf{f}',t)=\begin{matrix}(B_{s'}(B_{s'+1}(\dots B_{t-1}(f_t))),\dots,B_{t-1}(f_t),f_t,\\ F_t(f_t),\dots,F_{e'-1}(F_{e'-2}(\dots F_t(f_t))))\end{matrix}. \qquad (3)$$

The dissimilarity $d(\mathbf{f}',M'(\mathbf{f}',t))$ is defined as the mean geometric distance:

$$d(\mathbf{f}', M'(\mathbf{f}', t)) = \frac{\sqrt{\sum_{u=s'}^{e'} \left\| f_u - g_u \right\|^2}}{e'-s'+1}, \quad (4)$$

where $\mathbf{g}=[g_u]_{u=s'\ldots e'}$ stands for the warped trajectory $M'(\mathbf{f}',t)$.

Each warped trajectory being generated by composing motion models from the coordinates at a given reference instant, a trajectory obtained from erroneous coordinates would accumulate the errors, probably leading to a large dissimilarity with the query trajectory. That is why a minimization is performed in (2): to increase robustness, the minimum distance between the query trajectory and a possible warped trajectory given the affine trajectory model is retained, rather than relying on a single, possibly corrupted, warped trajectory.

III. Clustering Algorithm

Agglomerative hierarchical clustering algorithms (like [3]) are "bottom-up" approaches by which clusters progressively emerge in parallel. Such methods require the definition of a pairwise distance among input data points. In our problem, the definition of such a distance is somewhat restricted: based on two trajectories only, motion models with more than four parameters cannot be estimated and hence used in the definition of the distance.

Sequential "top-down" algorithms that determine one dominant cluster at a time are not undermined by previous problem. RANSAC [4] could for instance be applied sequentially: at each step a dominant motion model of possibly high dimensionality can be estimated and its outliers passed to the next iteration. The main drawback is that an inaccurate estimation of the first model may prevent the good detection of other motions.

Factorization methods of motion segmentation [7, 9, 12, 14] belong to partitional clustering algorithms that typically determine all clusters at once. In general, they use singular value decomposition (SVD) to factorize the feature trajectory matrix into a motion matrix and a structure matrix. The main drawback of such methods is that in presence of incomplete trajectories, extrapolation is required.

Recently, J-linkage [10] was proposed as a tailored agglomerative clustering algorithm. Although it was not applied to motion trajectories, it appears as an appealing tool to address our problem too. The idea is to combine the robustness of RANSAC for model estimation in the presence of outliers and pseudo-outliers and the flexibility of agglomerative hierarchical clustering. The clustering operates neither in the parameter space, nor in the residual space, but in the "conceptual" space where each data point is represented by the characteristic function of the set of models preferred by this data point. The number of clusters is determined automatically.

In order to adapt this clustering technique to our motion segmentation problem, we propose to use the motion-based residual defined in the previous section for a trajectory-model pair. We also propose a way to explicitly reject outliers before running the agglomerative process.

A. Variant of J-linkage

Since the estimation of each affine motion model in (1) requires at least three pairs of matched points at a given instant, we draw randomly minimal sample sets (MSS) of three point trajectories.

Unlike [10], during random sampling we do not encourage neighboring point trajectories to be selected with higher probability because an affine motion model is often more accurate when estimated from distant samples.

However, to handle the various life-spans of the trajectories, we impose that each one of the generated MSSs fulfills the following condition to be kept further. The common time section of the three sample trajectories must be composed with at least two instants (see Fig. 1). In other words, if $\mathbf{f}^{(1)}$, $\mathbf{f}^{(2)}$and $\mathbf{f}^{(3)}$ form a MSS, we must have:

$$\min(e(1), e(2), e(3))-\max(s(1), s(2), s(3)) > 0. \quad (5)$$

We generate N MSSs fulfilling this condition. An affine trajectory model estimated using a least squares method is associated to each MSS and its consensus set (CS) is computed. As in RANSAC, the CS is composed of all the inliers, i.e. all the point trajectories whose motion residual with the considered model is less than a threshold λ_1.

Then, as suggested in [10], for each point trajectory we build a preference set (PS) containing the indices of the models that the trajectory has given consensus to. Each point trajectory being represented by its PS, an agglomerative hierarchical clustering algorithm is performed. Initially, each PS forms a singleton cluster. Then, the algorithm merges successively the two closest clusters, using the Jaccard distance defined in (6) between PSs. After merging, the PS of the resulting cluster is computed as the intersection of the PSs of the merged clusters.

Given two PSs Q_1 and $Q_2 \subset \{1\ldots N\}$, the Jaccard distance measures the dissimilarity between the two sets as follows:

$$d_J(Q_1, Q_2) = \frac{|Q_1 \cup Q_2| - |Q_1 \cap Q_2|}{|Q_1 \cup Q_2|}, \quad (6)$$

where | . | denotes the cardinality of a set.

The merges carry on while the distance between the two closest clusters is strictly less than 1 (i.e. clusters with disjoint PSs cannot be merged). Note that the user has neither to provide the number of clusters to be extracted, nor to tune this cut-off value. Then, for each obtained cluster, the model is updated from the cluster inliers using a least squares method.

Finally, each point trajectory is re-assigned to the closest cluster according to the residual defined in (2).

Note that during models update, each model can be extended to every pair of consecutive instants for which we have at least three inliers trajectories (see Fig. 1).

B. Outliers Rejection

In [10] the authors assume that "outliers emerge as small clusters" and they propose to "reject all the smallest clusters up to the number of outliers" if this latter is known. In our application, however, the proportion of outliers is not known and is difficult to estimate. One may think of rejecting clusters whose cardinality is less than a threshold.

However it may happen in practice that an outlier is merged into a large cluster. It is the case if the PS of the outlier and the PS of the cluster are not disjoint, leading to a Jaccard distance less than 1. It is verified if it exists one model (one only is enough) the outlier and all the elements of the cluster have given consensus to. In general this kind of merging happens around the end of the agglomerative process. A cut-off value smaller than 1 could avoid such unexpected merges. This would however make other correct merges impossible, leading to too numerous clusters.

During the agglomerative process, as the merges follow one another, the cardinality of the PSs decreases (actually PSs are gradually reduced to models that fit all the cluster point trajectories) and the Jaccard distances between remaining clusters increase. Now, most of the outliers can be identified as having small PS or large Jaccard distances towards inliers. So we prefer to reject them even before running the agglomerative hierarchical clustering, by detecting that there is no other PS whose distance to the PS of the considered point trajectory is less than a threshold λ_2.

IV. Experimental Results

A. Results on the Hopkins155 Database

We tested our algorithm on the *Traffic* sequences and *Articulated* sequences of the Hopkins155 benchmark presented in [11][1]. All the reported trajectories are complete and there is no outlier. We set N=600 and λ_2=1 for all the experiments on this benchmark.

Since in our method the user does not provide the number of clusters to be extracted, we determined, with some hand-tuning on few sequences, 6 values of the parameter λ_1 that gave roughly the same numbers of clusters as the ones in the ground truth[2]. Then for each one of the 51 sequences and with each one of the 6 values of λ_1, we run the algorithm 100 times to average out the randomized part of our approach in the performance assessment.

In Table 1, we report the average and the median classification errors, as well as the average computation times of our algorithm, using for each sequence the values of λ_1 that made comparison possible. Statistics corresponding to the other algorithms come from [7]. This table shows that, on the test sequences and among the four compared algorithms, our method is in average the second most accurate, while being the fastest.

TABLE I. Misclassification rates and average computation times on the Hopkins155 database

Method	MSL [9]	LSA 4n [14]	ALCsp [7]	Our algo.
Traffic (2 motions) (31 sequences)				
Average	2.23%	5.43%	1.59%	1.92%
Median	0.00%	1.48%	1.17%	0.01%
Traffic (3 motions) (7 sequences)				
Average	1.80%	25.07%	7.75%	4.89%
Median	0.00%	23.79%	0.49%	0.19%
Articulated (2 motions) (11 sequences)				
Average	7.23%	4.10%	10.70%	5.38%
Median	0.00%	1.22%	0.95%	0.02%
Articulated (3 motions) (2 sequences)				
Average	2.71%	7.25%	21.08%	20.41%
Median	2.71%	7.25%	21.08%	20.41%
All 51 sequences				
Average	3.27%	7.91%	5.56%	3.80%
CPU	11h 24m	7.064s	15m 38s	5.734s

B. Results on Trajectories with Various Life-Spans

One of the distinctive features of our approach being its abilities to handle trajectories with various life-spans, we tested our algorithm on such types of trajectory sets.

For the sequences *Cars* and *Hand* (available online), we used the trajectories estimated in [8]. For the other sequences, the trajectories were extracted using the Kanade-Lucas-Tomasi feature tracker [1] with replacement of lost features to obtain trajectories starting after the first image.

Using the KLT tracker, one often obtains trajectories with very short life-spans corresponding to points quickly lost (e.g., because of occlusions, out-of-plane rotation, deformation, clutter). Also, when KLT is asked to detect and track a large number of points, a number of them might be in uniform regions with unstable trajectories as a consequence. Since our algorithm relies only on motion cue, all such short and/or unreliable trajectories are improper for clustering purpose.

As a consequence, and in order to assess precisely the clustering capabilities of our system in the absence of such nuisance, we retain only the P trajectories with the largest gradient on the start point and whose length is greater than a given minimum $l_{\min}$. Note however that all discarded trajectories can be classified posterior to the clustering step, by assignment to the closest cluster according to the residual defined in (2) as demonstrated in Subsection IV.C.

For every sequence except *Carmap*, we chose $l_{\min}$ large enough such that every random MSS fulfills (5), which allows to save some computation time.

For the *Carmap* sequence, we set $l_{\min}$ to a lower value such that trajectories corresponding to the car before and after occlusion by the map are not discarded.

[1] Our motion model is not appropriate to handle *Checkerboard* sequences because of the large rotations out of the image plane. For these sequences, we typically obtained more than three clusters, which prevented us to compare our results with the provided ground truth.

[2] Note that this selection of multiple λ_1 values based on cluster numbers is only conducted to make possible the comparison with clustering on this database by other methods where the number of clusters is prescribed.

For each sequence, we present the result of the clustering before models update and the final assignment after models update, on few frames. It explains why results of "clustering" and "final assignment" may be slightly different.

In *Cars* sequence (Fig. 2), it is interesting to note that even the few trajectories corresponding to the farthest car joined the cluster of the two other cars despite their smaller apparent motions. If the application would require it, the three cars could be easily separated using spatial prior.

Fig. 3 shows good results on the complex motions of the *Hand* sequence. Misclassifications are mainly due to trajectories that jump from the hand to the background (or conversely) during an occlusion. Such trajectories are not detected as outliers because they match very well the estimated motion models during not so short time intervals.

The ability to handle moving objects without any complete trajectory is illustrated by Fig. 4. Here, in the *Carmap* sequence, the car is partially occluded. The fact that the front re-appears before that the back disappears allows the extraction of a single motion cluster for the car before and after the occlusion, without requiring any trajectory completion.

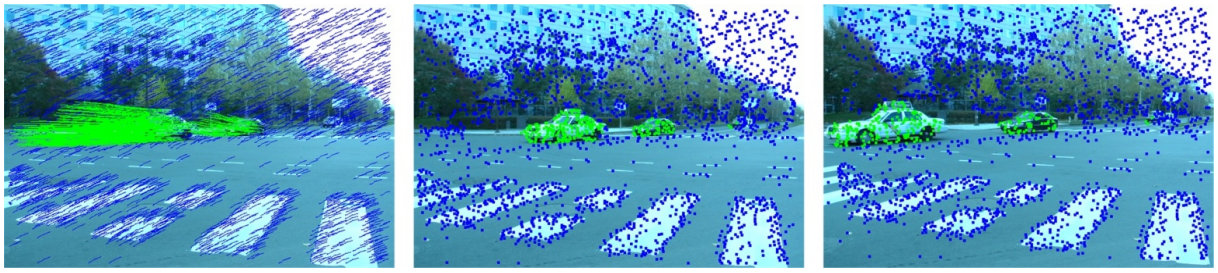

Figure 2. *Cars* sequence. (left) Automatic clustering of trajectories; (middle, right) final assignments displayed on trajectory points that are present in images 0 and 25 respectively.

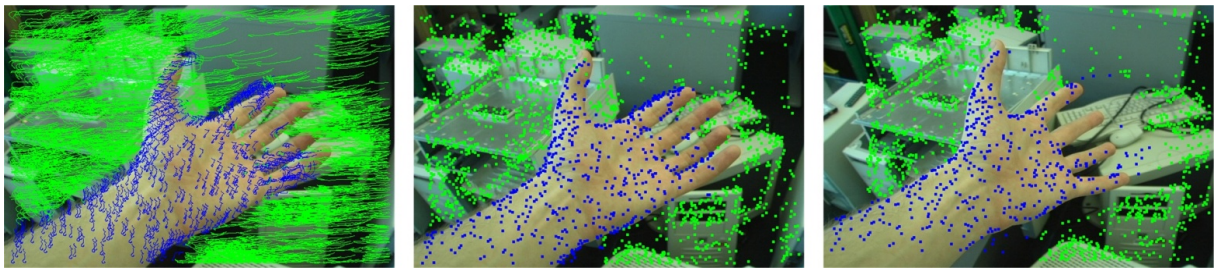

Figure 3. *Hand* sequence. (left) Automatic clustering of trajectories; (middle, right) Final assignments displayed on trajectory points that are present in images 0 and 35 respectively.

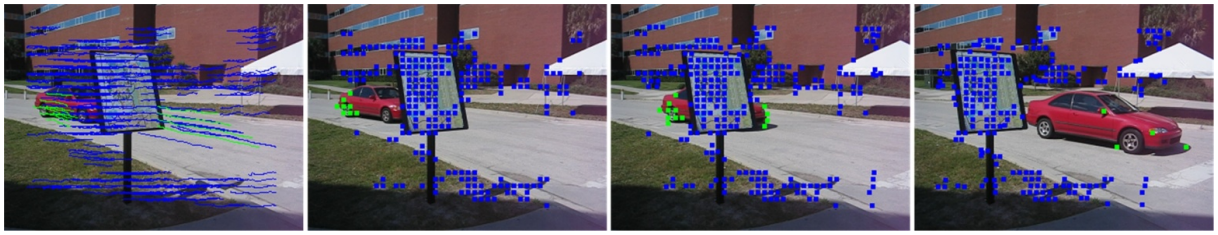

Figure 4. *Carmap* sequence. (from left to right) Automatic clustering of trajectories; Final assignments displayed on trajectory points that are present in images 0, 11 and 33 respectively.

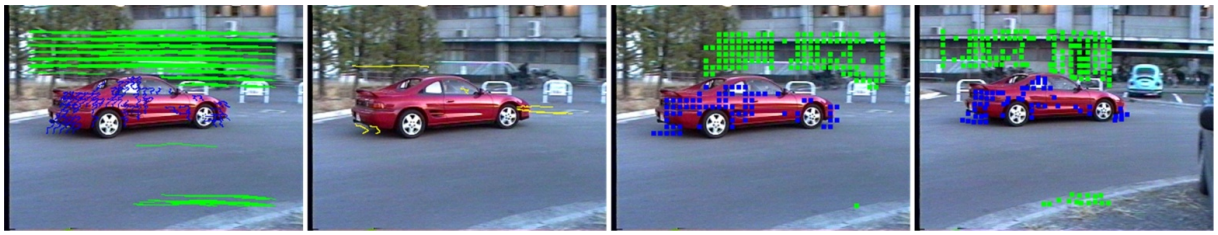

Figure 5. *Kanatani2* sequence. (from left to right) Automatic clustering of trajectories; Rejected outliers; Final assignments displayed on trajectory points that are present in images 0 and 16 respectively.

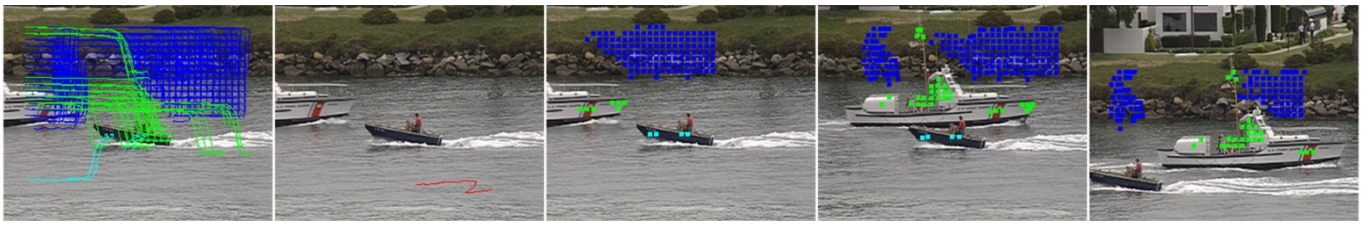

Figure 6. *Coastguarder* sequence. (from left to right) Automatic clustering of trajectories; Rejected outliers; Final assignments displayed on trajectory points that are present in images 0, 40 and 80 respectively.

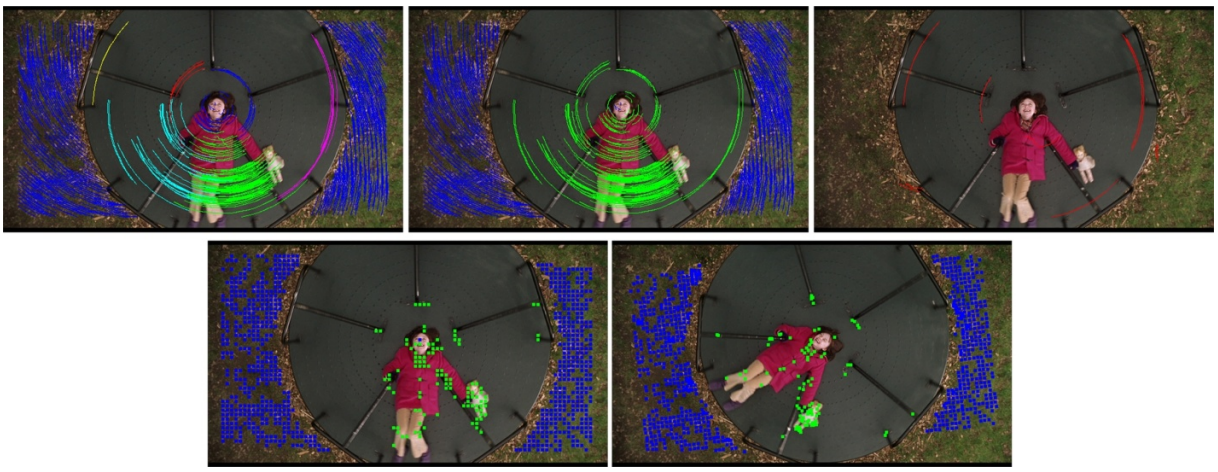

Figure 7. *Rotation* sequence. First row: (left) Agglomerative hierarchical method [3]; (middle) Our algorithm; (right) Outliers detected by our algorithm. Second row: Final assignments displayed on trajectory points that are present in images 0 and 21 respectively.

TABLE II. SEVERAL CHARACTERISTICS OF THE SEQUENCES AND TRAJECTORY SETS USED IN THE EXPERIMENTS, AND PARAMETERS USED TO PERFORM CLUSTERING OF THE TRAJECTORIES (SEE TEXT).

Sequence	F	P	$l_{\min}$	m	λ_1	λ_2	N	**#outliers**	**CPU**
Cars	26	2505	14	2.61%	3	0.7	600	0	112.312s
Hand	36	2285	20	6.95%	3	0.7	600	0	114.406s
Carmap	36	300	13	15.83%	1	0.7	1000	0	7.516s
Kanatani2	17	300	10	15.51%	1	0.7	600	7	1.859s
Coastguarder	81	221	50	17.76%	1	0.7	600	1	9.186s
Rotation	22	1000	15	4.34%	1	0.7	2000	9	63.796s

Fig. 5 shows our results on the *Kanatani2* sequence using 300 trajectories with various life-spans instead of the 63 trajectories provided by the Hopkins155 database. Seven outliers were rejected. In average, these outliers had given consensus to only 4% of the models estimated from the MSSs.

One outlier only was detected in the *Coastguarder* sequence (Fig. 6). This outlying trajectory corresponds to a point on the water for which tracking failed. It had given consensus to only 4% of the models.

As a comparison, we implemented the agglomerative method proposed by [3]. Fig. 7 shows that this method is unable to recover rotation motions even if it can merge into a same cluster trajectories that can be matched using a translation. By contrast, our method recovered satisfactorily the rotation motions.

For each sequence, we indicate in Table 2 the parameters values that we used and:

- the number of frames F,
- the size P of the trajectory set,
- the length $l_{\min}$ of the shortest trajectory present in the set,
- the percentage m of missing data:

$$m = \left(1 - \frac{\sum_{i=1}^{P}\left(e(i) - s(i) + 1\right)}{P \times F}\right) \times 100\,, \qquad (7)$$

- the number of detected outliers (#outliers),
- the computation time (excluding trajectories estimation) given that all the experiments were performed on a P-4 3.6GHz machine with unoptimized C++ code.

The number N of valid MSSs to be drawn is related to the repartition of the trajectories in the different clusters and to the percentage of outliers that are unknown and difficult to estimate in our case. Since N must be large enough so that a certain number (at least) of MSSs do not contain any outlier or pseudo-outlier, it could simply have been set to the same extremely high value for all the sequences. However it would lead to expensive computational time. Therefore we preferred to adapt it according to the complexity of the scene, which explains why it is greater for the sequences *Carmap* (large occlusion) and *Rotation*.

C. Posterior Classification of Ignored Trajectories

The point trajectories that were not used for clustering, whether they are too short or associated to insufficiently textured starting patches, can be simply classified posterior to the clustering step: each of them is assigned to the closest cluster according to the residual defined in (2).

Fig. 8-11 show such results of classification using motion cue only. All estimated trajectories are classified, from the ones with low gradient on the start point or very short life-span, to the ones with high gradient on the start point and life-span covering the whole sequence. Misclassifications mainly occur in the homogeneous regions. Such errors could probably be avoided by using a spatial prior or adding an appearance cue.

V. Conclusion

We presented a new algorithm for the clustering of point trajectories with various life-spans. We propose to model motions by time series of affine motion models estimated between consecutive instants. Motion models estimation, as well as motion residual computation, take into account the various life-spans of the trajectories. Combining "top-down" and "bottom-up" approaches, the presented clustering algorithm voluntarily does not exploit other image features than motion. Results show that our method efficiently clusters sets of trajectories without requiring the completion of the trajectories, especially those that correspond to points getting occluded or leaving the field of view.

A point that would deserve further investigation is the not so rare case of erroneous trajectories that start on an object and, as this object gets occluded, jump to the occluding object with the tracker not noticing it. Such trajectories are not outliers since they are composed of two shorter trajectories. The aim would be to identify such situations and to break the trajectory accordingly, before classifying its pieces.

References

[1] Stanley T. Birchfield, "KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker", 2007, http://www.ces.clemson.edu/stb/klt/.

[2] Gabriel J. Brostow and Roberto Cipolla, "Unsupervised bayesian detection of independent motion in crowds", Proc. Computer Vision Pattern Recognition, 2006.

[3] Dan Buzan, Stan Sclaroff, and George Kollios, "Extraction and clustering of motion trajectories in video", Proc. Int. Conf. Pattern Recognition, 2004.

[4] Martin A. Fischler and Robert C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", Communications of the ACM, vol. 24, 1981, pp. 381–395.

[5] Ce Liu, Antonio Torralba,William T. Freeman, Frédo Durand, and Edward H. Adelson, "Motion magnification", Proc. SIGGRAPH, 2005.

[6] Shrinivas J. Pundlik and Stanley T. Birchfield, "Real-time motion segmentation of sparse feature points at any speed", IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol. 38, 2008, pp. 731–742.

[7] Shankar Rao, Roberto Tron, René Vidal, and Yi Ma, "Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories", Proc. Computer Vision Pattern Recognition, 2008.

[8] P. Sand and S. Teller, "Particle video: long-range motion estimation using point trajectories", Proc. Computer Vision Pattern Recognition, 2006, http://rvsn.csail.mit.edu/pv/.

[9] Yasuyuki Sugaya and Kenichi Kanatani, "Geometric structure of degeneracy for multibody motion segmentation", Proc. Workshop on Statistical Methods in Video Processing, 2004.

[10] Roberto Toldo and Andrea Fusiello, "Robust multiple structures estimation with J-linkage", Proc. Euro. Conf. Computer Vision, 2008.

[11] Roberto Tron and René Vidal, "A benchmark for the comparison of 3-D motion segmentation algorithms", Proc. Computer Vision Pattern Recognition, 2007. http://vision.jhu.edu/data/hopkins155/.

[12] René Vidal and Richard Hartley, "Motion segmentation with missing data using Power-Factorization and GPCA", Proc. Computer Vision Pattern Recognition, 2004.

[13] Josh Wills, Sameer Agarwal, and Serge Belongie, "A feature-based approach for dense segmentation and estimation of large disparity motion", Int. J. Computer Vision, vol. 68, 2006, pp. 125–143.

[14] Jingyu Yan and Marc Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate", Proc. Euro. Conf. Computer Vision, 2006.

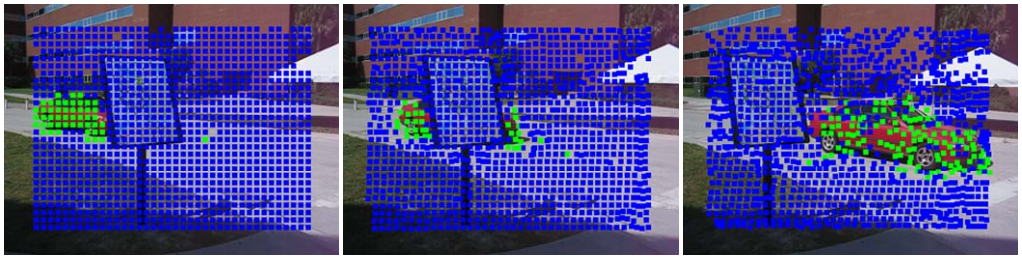

Figure 8. *Carmap* sequence. Classification of all the 2450 estimated trajectories whose lengths vary from 2 to 36. (from left to right) Final assignments displayed on trajectory points that are present in images 0, 11 and 33 respectively.

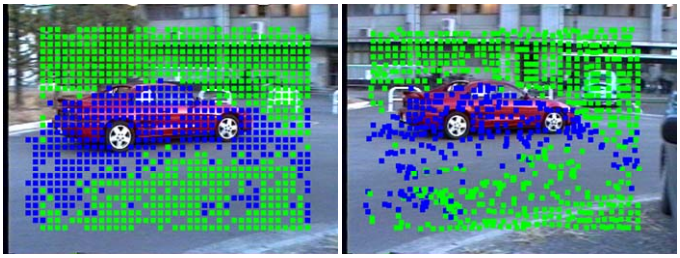

Figure 9. *Kanatani2* sequence. Classification of all the 3251 estimated trajectories whose lengths vary from 2 to 17. (from left to right) Final assignments displayed on trajectory points that are present in images 0 and 16 respectively.

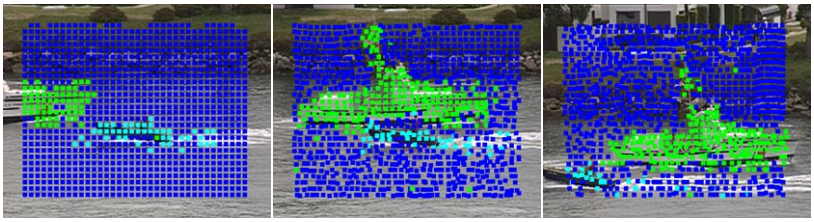

Figure 10. *Coastguarder* sequence. Classification of all the 6911 estimated trajectories whose lengths vary from 2 to 81. (from left to right) Final assignments displayed on trajectory points that are present in images 0, 40 and 80 respectively.

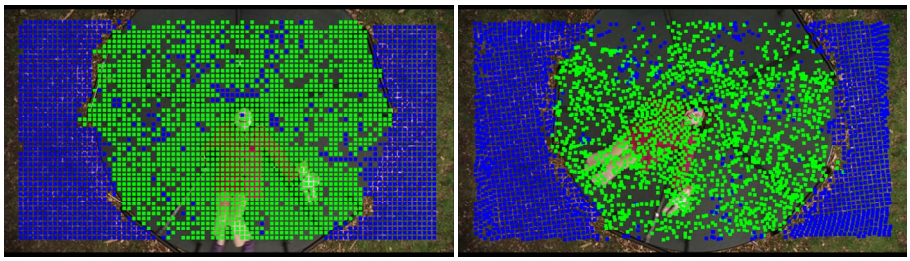

Figure 11. *Rotation* sequence. Classification of all the 12045 estimated trajectories whose lengths vary from 2 to 22. (from left to right) Final assignments displayed on trajectory points that are present in images 0 and 21 respectively.