SuperFloxels: A Mid-Level Representation for Video Sequences

Avinash Ravichandran[1], Chaohui Wang[1], Michalis Raptis[2] and Stefano Soatto[1]

[1] Vision Lab, UCLA
[2] Disney Research, Pittsburgh

Abstract. We describe an approach for grouping trajectories extracted from a video that preserves motion discontinuities due, for instance, to occlusions, but not color or intensity boundaries. Our method takes as input trajectories with variable length and onset time, and outputs a membership function as well as an indicator function denoting the *exemplar* trajectory of each group. This can be used for several applications such as compression, segmentation, and background removal.

1 Introduction

We are interested in establishing temporal correspondence in video, for the purpose of later analysis, for instance object or event detection, fine-scale localization, and recognition. In general, we seek a mid-level representation that would facilitate, or at least not jeopardize, semantic analysis, that is the attribution of identities and relations among locations or motions within a video.

Ideally, for every location in the domain of the image, we are interested in establishing its trajectory, or *flow*, in the time interval during which it is visible. Due to occlusions, we have to book-keep points that appear and disappear, and store the trajectory of each visible pixel, along with its color. Even if we neglect the color variability along a trajectory, it is easy to see that such a representation would soon have a complexity far greater than the original data. Therefore, to reduce complexity, we seek to group different trajectories. Since any grouping or segmentation procedure necessarily entails a loss of information, we seek to perform it in a way that causes as little damage as possible.

How do we measure damage? Ideally, by the loss in performance in the semantic analysis task downstream. This loss should be as small as possible. However, since we do not know the "ideal" performance, we can test directly for semantic consistency: If a certain set of locations is known to belong to an object or event at a certain time, we want the subsequent trajectories to also belong to the same object or event for all the time during which they are visible.

Since we do not have an end-to-end system available, our design criterion is to group together trajectories respecting motion discontinuities and occlusion boundaries, but not intensity/color boundaries. In analogy to *superpixels* that aim to preserve the latter, we call our scheme to preserve the former *superfloxels*.

Related Work The most common preprocessing operation performed on videos for the purpose of analysis is segmentation based on motion discontinuities, for the most part based on few temporally adjacent frames [1–3]. Trajectory-based methods have started to take hold more recently [4–10]. Naturally, the longer the temporal window, the richer the temporal context that can be incorporated in the later processing. This improves robustness and enables analysis of subtle cues for instance in traffic model learning [11] and anomaly detection [12].

There are, however, challenges associated with trajectory grouping. First, trajectories often cover only a portion of the frames in the video sequence, due to occlusions, image noise, *etc.* As a result, trajectories have different lengths and their comparison is not straightforward. Second, like any grouping procedure, there is no "right" number of clusters, and one has to accommodate different model complexities in the analysis. Finally, it is important but challenging to preserve occlusion boundaries so that clusters do not cross such boundaries.

Most existing trajectory-based methods ignore one or more of these issues. For example, [4, 5] assume that the number of clusters is known and tra- jectories are projected into different linear motion subspaces of low dimension using an affine camera model. Moreover, [4] requires complete trajectories, a tall order. Other approaches have been proposed to deal with incomplete trajectories, such as subspace separation [5, 6], extrapolation of incomplete trajectories [13], the adoption of Markov Random Field (MRF) formulation [10] as well as explicit clustering formulations [7–9]. Since the intrinsic complexity of the grouping method determines the scale of the problem it can handle, only sparse cases (*i.e*, tens to hundreds of trajectories) were considered in those factorization-based methods [4–6], while non-factorization-based methods [7–10] have been demonstrated with thousands of trajectories.

Among the non-factorization methods, [8] used spectral clustering and a post-processing procedure to deal with object segmentation. On the other hand, [9] used motion saliency to remove background trajectories and obtain the foreground map. A greedy search then partitioned the foreground trajectories based on their motion affinity and topology constraints. Despite promising results, the performance of such an approach relies heavily on the accuracy of the foreground map estimation and the greedy search process provides no optimality guarantee for the clustering. Finally, none except [10] explicitly accounts for occlusions. In the MRF-based formulation in [10], occlusions are modeled based on incomplete trajectories and "T-Junctions". However, discovering "T-Junctions" is still a challenging problem, that relies on local temporal information. Additionally, this method requires the number of clusters to be provided.

Paper Contribution In this paper, we describe an over-segmentation algorithm that clusters trajectories extracted from a video into *superfloxel*. Each *superfloxel* is denoted by a trajectory that is representative of the cluster of trajectories and serves as a mid-level representation of the video sequences. We achieve this by ensuring that our clustering scheme respects motion and occlusion boundaries and we influence the choice of the cluster centers using such cues.

Our approach is not an object segmentation scheme but instead a generic pre-processing framework that can be used for different applications such as activity recognition, object segmentation, *etc.* Our framework does not require multiple initializations or a predefined number of *superfloxels*. Instead, our method automatically determines the number of *superfloxels* by balancing the inter-cluster variation with the proposed prior terms on the cluster centers.

2 Formulation

Let $I : \mathcal{D} \subset \mathbb{R}^2 \times \mathbb{Z}^+ \rightarrow \mathbb{R}^+; (x,t) \mapsto I_t(x)$ be a video sequence, defined on a 2D image domain $\mathcal{D}$ and I_t denotes the image corresponding to frame t. We extract from I a set of trajectories $\mathcal{T} = \{T_i\}_{i=1}^N$, where each trajectory T_i is represented by $x_t^i \in \mathcal{D}$, where $t \in [t_s^i, t_e^i] \subset \mathbb{Z}^+$ denotes the temporal extent of the trajectory, starting at t_s^i and ending at t_e^i. Given these trajectories, we aim to cluster them into K groups (superfloxels) through a membership function, $L : \mathcal{T} \rightarrow [1, K];\ T_i \mapsto l_i$ where $K < N$. To this end, the first step is to define a notion of similarity between these trajectories on which we base our clustering.

2.1 Similarity between Trajectories

Given two trajectories T_i and T_j, we are interested in defining a measure of similarity between them. There are several choices for defining such metrics between trajectories such as the spatial distance or the velocity distance which can be defined as follows:

$$d_{spatial}(T_i, T_j)^2 = \sum_{f \in \mathcal{O}(T_i, T_j)} (x_f^i - x_f^j)^2, \tag{1}$$

$$d_{velocity}(T_i, T_j)^2 = \sum_{f, f+1 \in \mathcal{O}(T_i, T_j)} [(x_f^i - x_{f+1}^i) - (x_f^j - x_{f+1}^j)]^2, \tag{2}$$

where $\mathcal{O}(T_i, T_j) = [t_s^i, t_e^i] \cap [t_s^j, t_e^j]$ denotes the temporal overlap between T_i and T_j. Since our goal is to obtain a mid-level representation of the video, we require a grouping that is spatially compact and salient in its motion. Furthermore, we do not want the length of the trajectories to influence the pairwise distances. Considering all these requirements, we propose the following distance:

$$d(T_i, T_j) = \frac{1}{|\mathcal{O}(T_i, T_j)|} d_{spatial}(T_i, T_j)[1 + d_{velocity}(T_i, T_j)]. \tag{3}$$

We add the constant 1 to the velocity distance to ensure that when two trajectories are identical (motion wise), the spatial distance still acts as a weight. We wish to point out that different variations of the velocity distance have been used in the literature where the velocity difference is defined between the locations at time t and $t + t_0$ where $t_0 = 5$ in [8] $t_0 = 3$ in [9]. While this offset could make the distance more robust, manually choosing its value is difficult in practice as it depends on the frame-rate and the motion of objects in the scene.

The trade-off of using the above distance is that while we ensure spatial compactness of our superfloxels, we discount the velocity difference at motion

boundaries. Since motion boundaries provide strong cues for separating different objects/parts, we would like our distance to respect them while considering the similarity between two trajectories. Hence, we define the following distance that serves as a penalty for crossing motion boundaries:

$$d_{mb}(T_i, T_j) = \frac{1}{|\mathcal{O}(T_i, T_j)|} \sum_{f \in \mathcal{O}(T_i, T_j)} \int_0^1 B_f(x_f^i + \lambda(x_f^i - x_f^j))d\lambda, \tag{4}$$

where $B : \mathcal{D} \subset \mathbb{R}^2 \times \mathbb{Z}^+ \rightarrow \{0, 1\}$; $(x, t) \mapsto B_t(x)$ is the motion boundary at time t. This distance is added to Eq. 3 to yield our final similarity metric.

2.2 Clustering Formulation

In order to provide a useful reduction of data volume, we aim to group all the trajectories into a number of clusters, each called a *superfloxel*. Furthermore, we are interested in selecting an *exemplar* for each superfloxel such that it can best represent the properties of its components. If we assume that we know K, then in order to cluster the data points and obtain the cluster centers that are denoted by $\mathcal{C} = \{C_k\}_{k=1}^K \subset \mathcal{T}$, we minimize the objective function

$$\mathcal{E}(\mathbf{w}, \mathcal{C}) = \sum_{T_i \in \mathcal{T} \setminus \mathcal{C}} \sum_{C_k \in \mathcal{C}} w_{ik} d(T_i, C_k), \tag{5}$$

such that $w_{ik} \in \{0, 1\}$ and $\sum_k w_{ik} = 1$. Each binary variable w_{ik} indicates whether trajectory T_i is assigned to the cluster center C_k or not. Moreover, the constraint $\sum_k w_{ik} = 1$ guarantees that each trajectory will be assigned to exactly one cluster center. However, if we use the same function when K is unknown, the trivial solution to the above minimization is $K = |\mathcal{T}|$. In order to prevent this, we introduce a prior on the choice of cluster centers:

$$\mathcal{E}(\mathbf{w}, \mathcal{C}, K) = \sum_{C_k \in \mathcal{T}} \sum_{T_i \in \mathcal{T} \setminus C_k} w_{ik} d(T_i, C_k) + \sum_{C_k \in \mathcal{T}} w_{kk} \phi(C_k), \tag{6}$$

where the binary variable w_{kk} indicates whether trajectory C_k is a cluster center or not, defining implicitly the subset $\mathcal{C}$ of the cluster centers and leading the constraint on $\mathbf{w}$ that is $w_{ik} \leq w_{kk}$. The optimization problem of (6) is a Linear Integer program [14, 15], which can be efficiently solved via Linear Program (LP) relaxation. By relaxing the binary variables $\mathbf{w}$ to take non-negative values, the above formulation can be cast as an LP as shown in [14]. More specifically, our clustering formulation is expressed as the following LP:

$$\begin{aligned} &\min_{\mathbf{w}} \sum_{C_k \in \mathcal{T}} \sum_{T_i \in \mathcal{T} \setminus C_k} w_{ik} d(T_i, C_k) + \sum_{C_k \in \mathcal{T}} w_{kk} \phi(C_k) \\ &\text{subject to: } \sum_k w_{ik} = 1, w_{ik} \leq w_{kk}, w_{ik} \geq 0. \end{aligned} \tag{7}$$

The advantages of such an approach are: 1) it is guaranteed to converge; 2) it does not require initialization of the cluster centers or the number of cluster centers; and 3) we can specify the prior on the selection of the cluster centers. Our choice of prior $\phi(C_k)$ needs to reflect the fact that our clusters have low motion discrepancy and respect motion boundaries. A natural choice is to prevent trajectories that are near the motion boundaries to be chosen as cluster centers. We defer the discussion of the prior until Sect. 4.

3 Applications

The superfloxelization presented above forms our mid-level representation of the video sequence, based on which we can post-process the cluster trajectories for different applications. However, in all the cases, we only need to deal with the cluster centers of superfloxels. This has the following advantages: (1) the computational complexity of the post processing stage is significantly reduced, (2) we are robust to outliers that are present in the trajectories, by grouping the trajectories. We validate our representation by outlining several applications.

Model selection Our algorithm automatically selects the number of superfloxels. However, if the number of groups that are present in the video sequence is known, it can be enforced easily by merging superfloxels to the desired number. Given the cluster centers of the superfloxels, we calculate the same distance we use for clustering (Eq. 3). We drop the motion boundary distance since the cluster centers are spread throughout the image. Hence, calculating such a distance is not meaningful in this case. We can then obtain a hierarchical tree representation of these cluster centers via a greedy merging scheme using single linkage. At each step, the pair of superfloxel cluster centers (or group of superfloxels) that have the least distance among them are merged. This process is repeated until there is only one node in the tree (*e.g*, Fig. 2). Once two or more nodes have been merged, the distance between the merged node and any other node is calculated using single linkage: $d(C_i, C_j) = \min_{C_u \in C_i, C_v \in C_j} d(C_u, C_v)$. This tree can be cut to obtain the required number of groups. Our experiments show that this simple merging technique yields in good performance.

Background Removal For applications such as human activity recognition, trajectory-based methods have shown promising performance [16]. Hence in this case, the superfloxelization could be a useful preprocessing step in order to capture the different parts of the object as well as separate the foreground from the background. Recently, [16] showed that by removing the background trajectories, the performance of recognition algorithm increased on standard activity recognition benchmarks. Although methods such as [17] address the issue of removing the background from freely moving cameras, such methods used the entire set of trajectories that are present in video to determine the background. Hence, such methods are computationally expensive and do not provide a grouping of the remaining trajectories.

Given the cluster centers, we can exploit them to discard the background superfloxels. In order to perform this, for each pair of frames we fit a motion model based on the location of the points of the cluster centers in those frames using RANSAC [18], which determines the trajectories that are inliers and outliers. Repeating this for the different pairs of frames gives us the trajectories that are inliers and that are outliers. The background is then determined by the superfloxels corresponding to the cluster centers that were considered as inliers.

Fig. 1. Visualization of motion cues for a sample sequence. From left to right: original image, the optical flow field and the distance transform on motion boundaries.

4 Implementation Details

The spatial distance and the velocity distance computations are straightforward and can be done very efficiently. For the distance term in Eq. 4 that is defined over a straight line connecting two points, we speed up the computation by using an approximation to the straight line between two points (via Bresenham's algorithm [19]). Furthermore, calculating this distance over all pairs of points is not very meaningful. For instance, if two points that lie far apart in the background, the line between them could pass through motion boundaries corresponding to multiple objects. Hence, we calculate the motion boundary penalty only in a neighborhood around the motion boundaries. Using the distance transform [20] on the motion boundaries, we can quickly determine this neighborhood. Let $d_{B_f}(x)$ denote the distance transform under the motion boundaries B_f at a given time instance f. We calculate $d_{mb}(T_i, T_j)$ if $\forall f \in \mathcal{O}(T_i, T_j)\ d_{B_f}(x_f^i) \leq 2\tau$ and $d_{B_f}(x_f^j) \leq 2\tau$. The parameter τ that determines the neighborhood around the motion boundaries for which we calculate d_{mb} (Fig. 1).

Notice from Fig. 1 that points close to the edge of the image tend to have a high value. Hence, if we use our prior based on the distance transform, such points will have a higher likelihood of being chosen as cluster centers. In order to prevent this and to ensure that our prior is consistent with our distances between trajectories, we define the prior on the clusters centers as follows:

$$\phi(T_i) = \begin{cases} \gamma & \text{if } \exists f \in [t_s^i, t_e^i] \text{ s.t. } d_{B_f}(x_f^i) < \tau \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

This biases cluster centers to be chosen further from motion boundaries.

5 Experimental Results

We extracted trajectories using the Large Deformation Optical Flow (LDOF) [21] and the Dense Point Tracker (DPT) [22]. We also experimented with the Sparse Occlusion Optical Flow (SOOF) [23] and DPT to obtain the trajectories. This was motivated by the fact that [23] accounts explicitly for occlusions while calculating the optical flow. Hence, trajectories that are near the occlusion

Table 1. Comparison of Different Algorithms using Different Metrics.

Method	Density	Overall Error	Average Error	Over segmentation	# of extracted objects
[8]	3.316	4.861	25.870	0.692	26
[8] + SOOF	3.155	4.961	24.344	0.654	28
Our Method + LDOF	3.316	3.675	**10.175**	27.654	30
Our Method + SOOF	3.155	**2.880**	10.586	30.885	28
Background Merge + LDOF	3.316	7.284	21.971	7.269	27
Background Merge + SOOF	3.155	6.822	23.577	8.538	23

Table 2. Comparison of Different Algorithms using the Trimmed Mean of Metrics.

Method	Density	Overall Error	Average Error	Over segmentation	# of extracted objects
[9]	3.22	3.760	22.06	1.150	25
Our Method + LDOF	3.313	3.468	**9.580**	27.500	30
Our Method + SOOF	3.158	**2.510**	10.311	30.833	28
Background Merge + LDOF	3.313	5.711	21.174	6.950	27
Background Merge + SOOF	3.158	4.931	22.763	8.167	23

edges of the moving object can be better tracked. We tested our method on the MOSEG Database [8] which contains 26 sequences of variable length. Each sequence contains a few frames with ground truth annotation. Additionally, this database comes with its own evaluation software for quantitative analysis. We use the same experimental setup as [9], *i.e* we apply our algorithm to the first 50 frames of each sequence and if the sequence contains less than 50 frames we use the entire video sequence. In all our experiments we set $\tau = \gamma = 20$.

Quantitative Results We report the quantitative results for the different variations of our algorithm as well as that of the baseline [8]. We calculated different metrics such as the overall error, the average error, the over-segmentation index as well as the number of objects that are extracted. The overall error determines if a trajectory is assigned to the correct label. If a cluster spans multiple objects, then the points on one of the objects are considered as errors while determining the overall error. The average error determines the mean error over each region as opposed to the entire image area. The number of objects extracted are regions that have less than 10% error in their clustering assignment. Finally, the over-segmentation determines the number of groups that need to be merged to get the ground truth annotation. The evaluation tool also calculates density which is the measure of the percentage of points that are labeled in the video.

The quantitative results are shown in Table 1. Here we report the mean over all the sequences in the database for the different variations of both the baseline algorithm as well as our methods. From this table it can be seen that the baseline using SOOF produces better results compared to using LDOF. Our method of superfloxelization is better when compared to the baseline in all metrics.

The comparison with [9] is reported in Table 2. Notice that the numbers in this table are different from that of Table 1, because [9] uses a trimmed mean as

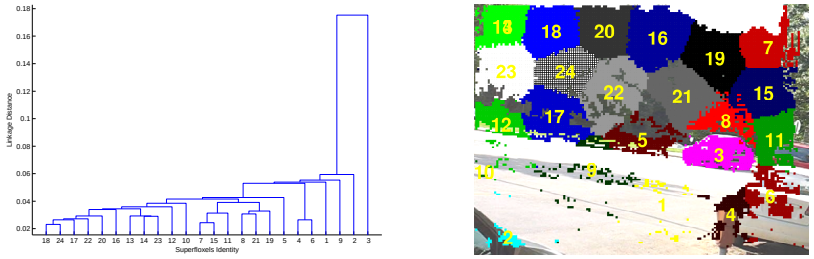
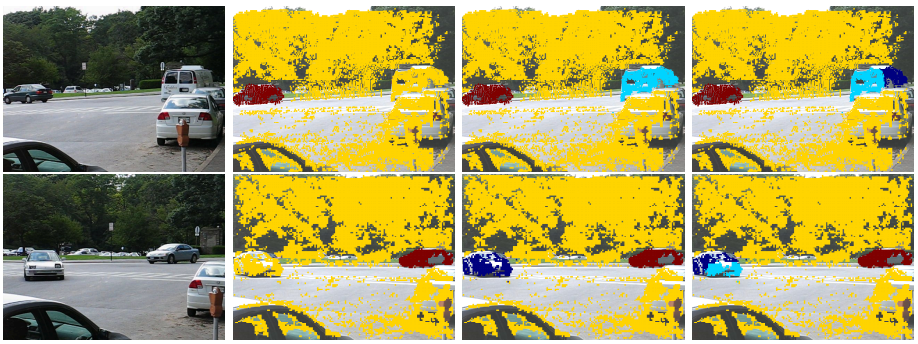


Fig. 2. The superfloxel tree structure obtained using the merging procedure (left) and the superfloxel image locations with numbers indicating their indices (right).

measures (the top 10% and the bottom 10% of the values are thrown out before computing the mean). Although we do not endorse this evaluation methodology, for the sake of comparison we report the trimmed mean of our results as well.

Choosing the number of superfloxels We earlier outlined how we can build a tree representation of the superfloxels. The tree structure for one video is shown in Fig. 2. From this figure we can see that the tree structure captures the overall structure of the scene. Note that this tree is only obtained using the cluster centers of the superfloxels. This demonstrates that our cluster centers are good representative candidates of the whole superfloxel. When $K = 2$, the cut of the tree gives us node 3 which corresponds to the entire car as one cluster and the rest of the nodes as the other cluster. We show other examples of choosing the value of K in Fig. 3, which shows that as we increase K the different objects that are present in the scene appear. Once the number of superfloxels that are required becomes more than the number of objects, we start to observe over-segmentation. The advantage of having the superfloxels is that we can now vary K as we like without clustering the entire set of trajectories again.

Background removal We have outlined a method to remove the background and show sample results in Fig. 4, which illustrates that our method is able to successfully remove the background in these videos. For a video of 50 frames that contains 16000 trajectories, the background removal after superfloxelization takes about 2 seconds using an unoptimized Matlab implementation. A quantitative analysis was also performed; we determine the background superfloxels and merge it as one group and then apply the quantitative evaluation used for the original superfloxels. The quantitative results are presented in Table 1, which show that while the overall error is increased by this process, the average error and the number of objects is better than the baseline. We notice that using the SOOF optical flow resulted in smaller number of extracted objects. Also, the number of segments that are needed to be merged to obtain the objects is much lower. In all our experiments, we used a homography as the motion model between two frames.



(a) Original Image (b) K=2 (c) K=3 (d) K=4

Fig. 3. Choosing the number of superfloxels using our tree representation: (a) Original Image; (b, c, d) show the different superfloxels for varying values of K.

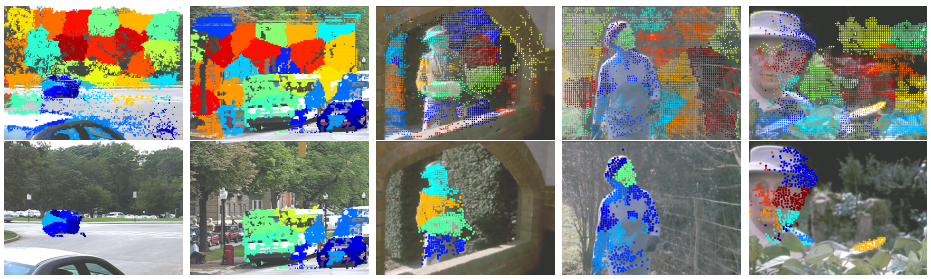


Fig. 4. Example of background removal. Top row shows the resulting superfloxels, Bottom row shows the background removed using the cluster centers of the superfloxels. Notice that in the last column, the leaf that is moving is not considered as background.

6 Conclusion

Superfloxels are agglomeration of trajectories in video that preserve motion discontinuities. They are designed to be a generic preprocessing step for video analysis. We have shown some sample applications of our framework and its performance. Our approach enables accommodating tracks of different lengths, and produces as a result a collection of groups of tracks that can be further refined in the presence of prior knowledge, for instance on the number of objects in the scene or other prior models. Quantitative comparison on benchmark datasets shows that our method outperforms other methods for trajectory clustering.

Acknowledgments This work was supported by NSF CCF-0969032, ARO W911NF-11-1-0391 and ONR N000141110863. We thank Nikos Komodakis and Nikos Paragios for providing the clustering code. We also thank Alper Ayvaci for valuable discussions.

References

1. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI **22**(8) (2000)
2. Kumar, M.P., Torr, P.H.S., Zisserman, A.: Learning layered motion segmentation of video. In: ICCV. (2005)
3. Cremers, D., Soatto, S.: Motion competition: a variational approach to piecewise parametric motion segmentation. IJCV **62**(3) (May 2005) 249–265
4. Yan, J., Pollefeys, M.: A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In: ECCV. (2006)
5. Vidal, R., Hartley, R.: Motion segmentation with missing data using powerfactorization and GPCA. In: CVPR. (2004)
6. Rao, S., Tron, R., Vidal, R., Ma, Y.: Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In: CVPR. (2008)
7. Fradet, M., Robert, P., Perez, P.: Clustering point trajectories with various lifespans. In: CVMP. (2009)
8. Brox, T., Malik, J.: Object segmentation by long term analysis of point trajectories. In: ECCV. (2010)
9. Fragkiadaki, K., Shi, J.: Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement. In: CVPR. (2011)
10. Lezama, J., Alahari, K., Sivic, J., Laptev, I.: Track to the future: Spatio-temporal video segmentation with long-range motion cues. In: CVPR. (2011)
11. Wang, X., Tieu, K., Grimson, E.: Learning semantic scene models by trajectory analysis. In: ECCV. (2006)
12. Piciarelli, C., Micheloni, C., Foresti, G.L.: Trajectory-based anomalous event detection. IEEE TCSVT **18**(11) (2008)
13. Brostow, G.J., Cipolla, R.: Unsupervised bayesian detection of independent motion in crowds. In: CVPR. (2006)
14. Komodakis, N., Paragios, N., Tziritas, G.: Clustering via LP-based stabilities. In: NIPS. (2009)
15. Charikar, M., Guha, S., Tardos, É., Shmoys, D.: A constant-factor approximation algorithm for the k-median problem. JCSS **65**(1) (2002)
16. Wu, S., Oreifej, O., Shah, M.: Action recognition in videos acquired by a moving camera using motion decomposition of lagrangian particle trajectories. In: ICCV. (2011)
17. Sheikh, Y., Javed, O., Kanade, T.: Background subtraction for freely moving cameras. In: ICCV. (2009)
18. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM (6) (1981)
19. Bresenham, J.E.: Algorithm for computer control of a digital plotter. IBM Systems Journal **4**(1) (1965)
20. Felzenszwalb, P.F., Huttenlocher, D.P.: Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science (2004)
21. Brox, T., Malik, J.: Large displacement optical flow: Descriptor matching in variational motion estimation. PAMI **33**(3) (2011)
22. Sundaram, N., Brox, T., Keutzer, K.: Dense point trajectories by gpu-accelerated large displacement optical flow. In: ECCV. (2010)
23. Ayvaci, A., Raptis, M., Soatto, S.: Sparse occlusion detection with optical flow. IJCV (2011)