Recognizing activities with cluster-trees of tracklets

Adrien Gaidon, Zaid Harchaoui, Cordelia Schmid

Recognizing activities with cluster-trees of tracklets

Adrien Gaidon
http://lear.inrialpes.fr/people/gaidon
Zaid Harchaoui
http://lear.inrialpes.fr/people/harchaoui
Cordelia Schmid
http://lear.inrialpes.fr/people/schmid

LEAR - INRIA Grenoble, LJK
655, avenue de l'Europe
38330 Montbonnot, France

Abstract

We address the problem of recognizing complex activities, such as pole vaulting, which are characterized by the composition of a large and variable number of different spatio-temporal parts. We represent a video as a hierarchy of mid-level motion components. This hierarchy is a data-driven decomposition specific to each video. We introduce a divisive clustering algorithm that can efficiently extract a hierarchy over a large number of local trajectories. We use this structure to represent a video as an unordered binary tree. This tree is modeled by nested histograms of local motion features. We provide an efficient positive definite kernel that computes the structural and visual similarity of two tree decompositions by relying on models of their edges. Contrary to most approaches based on action decompositions, we propose to use the full hierarchical action structure instead of selecting a small fixed number of parts. We present experimental results on two recent challenging benchmarks that focus on complex activities and show that our kernel on per-video hierarchies allows to efficiently discriminate between complex activities sharing common action parts. Our approach improves over the state of the art, including unstructured activity models, baselines using other motion decomposition algorithms, graph matching, and latent models explicitly selecting a fixed number of parts.

1 Introduction

Video content often relates to humans and their actions. Oft-sudied simple actions like running rarely convey enough meaning to interpret a scene, but instead constitute the building blocks of higher-level activities such as pole vaulting. These more interesting and complex activities are spatio-temporal patterns composed of several related movements of actors, their body parts, and objects. Automatically identifying those parts and leveraging the resulting structure is a challenging problem that is crucial for the recognition of activities. In this paper, we refine video comparisons using data-driven parts and their relations. We propose a large-scale unsupervised approach to hierarchically decompose the motion content of a video and introduce an efficient algorithm to compare two tree-structured videos.

Our approach relies on local point trajectories corresponding to pixels tracked across a fixed small number of frames: *tracklets*. They combine useful aspects of both point trajectories and local features. Similar to trajectories [1, 6, 20, 29, 32, 35, 38, 44], tracklets differentiate the time domain from the spatial one. They provide a structured and information-rich

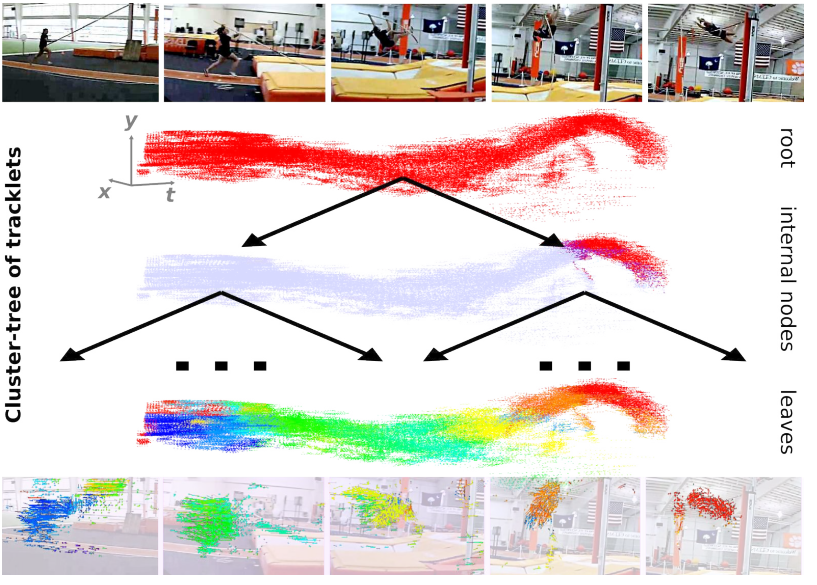

Figure 1: Example of a hierarchical motion decomposition obtained by our divisive clustering algorithm. We, first, extract dense tracklets to represent the motion content of a video. We, then, recursively bi-partition this set (the root of the cluster-tree) until the nodes reach a minimal size or cannot be split in a way that decreases a connectivity cost. The leaves, the internal nodes, and the parent-child relations all contain relevant information. Our approach uses the whole cluster-tree to model videos.

characterization of the motion of underlying parts. Similar to local features [3, 9, 22, 48, 51], tracklets can be easily and reliably extracted [49]. Their short duration also limits drifting problems, *i.e.* trajectories deviating from the underlying tracked object. This is in contrast to long-range trajectories [5, 23, 40], which are expensive to compute and face difficulties in capturing fast and articulated motions due to their sensitivity to occlusions. In addition, tracklets can be represented by adapting local descriptors such as Motion Boundary Histograms (MBH) [8, 49]. Recently, models based on tracklets [16, 29, 30, 49] have shown promising results on challenging video data like the Youtube [24] and Hollywood benchmarks [21, 28]. These datasets, however, focus on relatively simple actions — *e.g.* running or answering a phone — and, as we show in our experiments, more complex activities — *e.g.* in sports videos [34] — require more structured methods than the aforementioned ones.

Therefore, we introduce a novel hierarchical model more suited to the representation of the complex content of activities. Our method consists in, first, extracting tracklets over a dense spatio-temporal grid. Second, in contrast to most trajectory-based methods, we represent each tracklet with *two separate sets of features*: trajectory descriptors used for intra-video clustering, and the more robust optical-flow based MBH used for inter-video comparisons. Third, we decompose the motion content of a video by hierarchical *divisive* clustering on its tracklets, thus representing it as a *hierarchy of data-driven parts*. Our approach is *weakly supervised*: it relies neither on part annotations, nor on a predefined action decomposition such as [15]. In addition, we use neither expensive video segmentation techniques [4, 5, 17, 23, 31, 45, 47, 52], nor pre-trained object detectors [12, 37]. Although our tracklets are of a short constant duration, internal nodes in our hierarchical decomposition can capture complex and long-term motions of spatio-temporal parts, as well as their rela-

tions. Note, however, that these parts do not necessarily correspond to body parts or objects as our decomposition is data-driven and unsupervised.

Our first contribution is a hierarchical spectral clustering algorithm, based on top-down recursive bi-partitioning. Contrary to most spectral clustering approaches [13], we do not rely on a distortion minimization algorithm like k-means [18]. Instead, we minimize a spatio-temporal *connectivity* cost that allows for clusters of arbitrary shape and an efficient greedy splitting strategy that automatically determines the number of clusters. In contrast to bottom-up agglomerative algorithms — which have a computational complexity that is at least quadratic in the number of points [14, 18] — our *divisive* method can scale to videos with hundreds of thousands of tracklets. The resulting hierarchical decomposition provides structural information relating motion parts together.

Our second contribution is the use of this entire tree structure, called *cluster-tree* [10] (*c.f.* Figure 1), in order to build a hierarchical model of the motion content of a video. This is in contrast to existing approaches [39] that view videos as a bag of clusters. We introduce a corresponding tree representation of actions, called *BOF-tree*. The BOF-tree of a video has the same structure as its cluster-tree and each node is modeled by a bag-of-features (BOF) over the MBH descriptors [49] of its constitutive tracklets. Efficiently using this structural information is challenging as cluster-trees have a variable number of nodes and a structure specific to each video. Furthermore, there is no natural left-to-right ordering of the two children of a parent node. Therefore, we introduce an efficient kernel on variable-size, unordered binary trees. We use it in conjunction with powerful non-linear SVM classifiers on videos represented by BOF-trees, *i.e.* hierarchically structured sets of motion components.

Related work. Niebles *et al.* [33] propose to represent an action as a constellation of parts represented by BOFs over shape and motion features. They model a category using a probabilistic mixture of a fixed number of parts. They classify actions in controlled video conditions by maximizing the likelihood with respect to their generative model. Brendel and Todorovic [4] introduce a more general graphical model learned from hierarchical video segmentations. Their approach assumes that all actions of the same category share strong geometrical and temporal part relationships. Furthermore, their video segmentation algorithm can only account for smooth motions of color-consistent parts.

Discriminative alternatives to these generative models are often based on the popular deformable part model of Felzenszwalb *et al.* [12]. For instance, Liu *et al.* [25] combine manually predefined attributes with data-driven ones obtained by clustering local features. They use a latent SVM [12] to learn the importance of each part. Wang and Mori [50] use tracking and a Hidden Conditional Random Field (HCRF) to learn a discriminative model of latent parts for frame-by-frame recognition. Closest to our work, Raptis *et al.* [39] extract clusters of long-term trajectories and learn a latent model over a fixed number of parts. Their approach has a cubic time complexity in the number of trajectories, relies on bounding box annotations, and uses only a fixed small subset of clusters for all videos. Furthermore, they explicitly model pairwise relationships between clusters using their mean group trajectory, whereas we use the full hierarchical structure between mid-level components that results from our clustering.

In contrast to all the aforementioned discriminative approaches, we do not assume that actions share a fixed number of parts common to all training instances. Each video has its own decomposition structure and all parts — including their relationships — are used in our video comparisons. In addition, as we do not rely on latent parts, we do not need to solve a complex inference problem for each test video.

2 Clustering dense tracklets

In this section, we address the problem of efficiently clustering a video composed of a large number of tracklets in order to obtain a hierarchical decomposition of its motion components. We describe the two steps of our clustering algorithm: a non-linear projection using multiple tracklet features, followed by a hierarchical divisive clustering algorithm. Note, that the algorithm described in this section is applied to a single video at a time and does not require any other external data, such as other videos containing the same action.

2.1 Dense tracklets for intra-video clustering

We follow the approach described in [49] in order to efficiently track densely sampled points. We first compute the dense optical flow field of the video using Farnebäck's algorithm [11] as implemented in the OpenCV library [2]. We then track densely sampled pixels by motion interpolation with a median filter. We use a sampling step-size of 5 pixels along both the x-axis and the y-axis and filter out points in homogeneous regions. We track points across $L = 5$ frames in order to limit drift and have fixed length short trajectories. We observed that a longer track length cannot account for fast motions and yields more erroneous tracks due to motion blur and self-occlusions. On the other hand, tracklets shorter than 5 frames do not contain enough information to build discriminative models.

We model tracklets using multiple features describing both their spatio-temporal position and shape. We use the following descriptors to represent trajectory information: (i) x positions over time $\mathbf{x} = (x_1, \cdots, x_L)$, (ii) y positions over time $\mathbf{y} = (y_1, \cdots, y_L)$, (iii) temporal positions $\mathbf{z} = (t, \cdots, t+L-1)$, (iv) velocities along the x-axis $\mathbf{v_x} = (x_{k+1} - x_k)_{k=1:L-1}$, (v) velocities along the y-axis $\mathbf{v_y} = (y_{k+1} - y_k)_{k=1:L-1}$. Note, that our descriptors need not be particularly robust as all comparisons are *intra-video* during the clustering stage. We also separate the trajectory information along the different spatio-temporal dimensions in order to allow for dimension-specific normalizations in the later stages of our algorithm.

Our approach is not specific to the set of features chosen to represent a tracklet. The only pre-requisite is the availability of a similarity function specific to each feature channel. We use Gaussian RBF kernels $k(f, f') = exp(-\gamma d(f, f')^2)$, where the distance $d(f, f')$ is the Euclidean distance. The γ parameters of the kernels are automatically fixed to $\gamma = 1/(2\bar{d})$, where $\bar{d}$ is an estimate of the median of the distances between tracklets. This normalization ensures that all kernels are comparable across the different feature channels.

2.2 Multi-modal spectral embedding of tracklets

The first step of our clustering algorithm is to compute a spectral embedding of the tracklets. This projection relies on spectral properties of a similarity matrix between tracklets. As we have multiple features representing different characteristics of tracklets, we use as similarity the *product* of the per-feature similarities. As each similarity is positive-definite, the product similarity is also a positive-definite kernel. In contrast to concatenating the descriptors into a single vector or summing the per-feature kernels ("early fusion"), using this product kernel has the advantage of entailing an "and" effect: tracklets close according to this similarity are close with respect to *all* the features used.

The similarity matrix $W \in \mathbb{R}^{N \times N}$, between N tracklets, can be viewed as the weighted adjacency matrix of a graph, where the nodes are the tracklets and the edges are weighted by their pairwise affinity. To compute the spectral embedding of our tracklets, we project

them onto the leading eigenvectors of the normalized Laplacian of this similarity graph: $\mathcal{L} = I - D^{-1/2}WD^{-1/2}$, where D is the diagonal matrix of row-sums of W. Thresholding the eigenvector with the second smallest eigenvalue of $\mathcal{L}$ yields an approximate solution to the NP-Hard Normalized Cut (NCut) bi-partitioning problem [46]. As the number of tracklets N can go up to 10^6, we efficiently compute our spectral embedding using the Nyström method [13, 41]. It approximates the eigenvectors of the full similarity matrix W by just computing a small subset $n \ll N$ of its columns. In our experiments, we use $n = 1000$. The computational cost of this approach is in $O(n^2N)$ time and $O(nN)$ space, which is a large improvement over the initial $O(N^3)$ time, $O(N^2)$ space complexity.

2.3 Hierarchical divisive clustering

Using the computed spectral embedding, we cluster tracklets via an efficient hierarchical divisive algorithm. Existing divisive clustering methods often consist in repeatedly applying a standard clustering algorithm (*e.g.* k-means [18]). We found that, for our problem, this approach often yielded too imbalanced bi-partitions with almost empty or too large clusters. Instead, we propose to recursively bi-partition nodes by thresholding along an eigenvector, *i.e.* along a dimension of the spectral embedding. As the second smallest eigenvector is the real-valued solution to the NCut problem, it is composed of two clearly separated ranges of values that indicate the optimal partition of the tracklets. This is also valid for the next leading eigenvectors and, theoretically, one can optimally sub-partition the data by recursively thresholding one eigenvector after the other. In practice, however, the optimal bi-partition is often unclear and the eigenvectors tend to reflect this ambiguity by having smooth variations.

Therefore, we propose to split along an eigenvector by selecting the best threshold according to a spatio-temporal *connectivity* criterion. We associate to each node a cost that is its number of connected components in a spatio-temporal neighbourhood graph. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph of reciprocal k-nearest-neighbours over all tracklets from a video. An undirected edge (i, j) is in $\mathcal{E}$ if the tracklets $i, j \in \mathcal{V}$ are spatio-temporal k-nearest-neighbours. This graph is efficiently precomputed only once (before clustering) using kd-trees on the average spatio-temporal positions of each tracklet. We select the smallest k such that the entire video has exactly one spatio-temporally connected component. Due to the density of our tracklets, this yields approximatively $k = 10$ neighbours on average over all videos. The graph $\mathcal{G}$ is, therefore, sparse. Computing the cost of a node — *i.e.* a set of tracklets $\mathcal{V}' \subset \mathcal{V}$ — is then done by counting the number of connected components in the subgraph $\mathcal{G}(\mathcal{V}')$ using a simple depth-first search. In contrast to distortion-based costs such as inertia, this connectedness criterion does not constrain the shape of the clusters. It only penalizes nodes with spatio-temporally disjoint parts.

Starting from the root node containing all tracklets of a video, we build the cluster-tree using a greedy top-down approach. We search over nine adaptive thresholds for each split: the 10^{th}, 20^{th}, ..., and 90^{th} percentiles of the considered eigenvector values. If the best split along this eigenvector yields children nodes with a lower cost than their parent node, then (i) the split is registered with this eigenvector-threshold pair, (ii) the node is marked as processed, and (iii) its two children nodes are added to the cluster-tree. Otherwise, we iteratively try to split the current node along the next leading eigenvector, until we reached the last one. If an improving split was not found, the node is a leaf and is marked as processed. We then repeat these steps on the node with the highest cost that was not processed yet and iterate until all nodes of the cluster-tree have been processed. Note, that the root is given an infinite cost in order to force the first split. For each new node to be processed, we start

again from the leading eigenvector, as suggested in [46], because it is the most reliable one. We also observed that, in practice, the leading eigenvectors are often composed of several approximately flat plateaus.

3 Classification of cluster-trees

In this section, we first explain how we represent each node in the cluster-tree as a histogram of quantized tracklet features. We then describe how we perform action recognition by efficiently comparing cluster-trees using an adapted kernel.

3.1 Tracklet description for classification

Contrary to the video-specific clustering step, recognition requires a model that can handle the large intra-class variability of activities in real-world videos. Therefore, we represent a tracklet using Motion Boundary Histograms (MBH) [49]. This robust local motion descriptor is analogous in spirit to the well-known SIFT descriptor [26]. It consists of two histograms quantifying the gradients of the horizontal and vertical components of the optical flow. It robustly handles the noise in the spatial derivatives of the optical flow *via* quantization. We use the same parameters as in [49] to efficiently compute tracklet-aligned MBH descriptors directly from the dense optical flow field used to obtain the tracklets. This local descriptor is particularly suited to represent tracklets in the context of action classification with a simple bag-of-features (BOF) video model [49].

3.2 BOF-Tree: tree of nested bag-of-features

From the set of MBH descriptors and the cluster-tree obtained by our spectral divisive algorithm, we extract a hierarchical representation of a video called *BOF-tree*. Its structure is the same as its corresponding cluster-tree. In addition, each node in the BOF-tree is modelled by a BOF, *i.e.* a sparse histogram over its quantized MBH features. We first pre-compute a vocabulary of track-aligned MBH features on a random subset of the training features. We use an on-line k-means algorithm [42] with $k = 4000$ to cluster 10^6 tracklets randomly sampled from the training videos. We then quantify all MBH features by assigning them to the closest "visual word" (centroid) in the vocabulary. Finally, each node is represented by its histogram of occurrences of visual words. Modelling a node with a BOF disregards the unreliable shape of a node (which may contain spurious tracklets), while local geometry is still captured by the tracklet features, and global structure information is captured by the cluster-tree.

As a consequence of the clustering, the left-to-right order in the BOF-tree does not have a direct geometrical interpretation (the cluster-tree is *unordered*). The only semantic relation directly captured by our tree structure is the inclusion relation derived from the cluster-tree: the two children of a node correspond to a bi-partition of the tracklets of this parent node. This induces an additive property on the nodes of a BOF-tree: the BOF of a node in a BOF-tree is the sum of its children's BOFs. See Figure 2 for an illustration. We now show how to use this property to efficiently compare BOF-trees, while leveraging the hierarchical structure information between nodes.

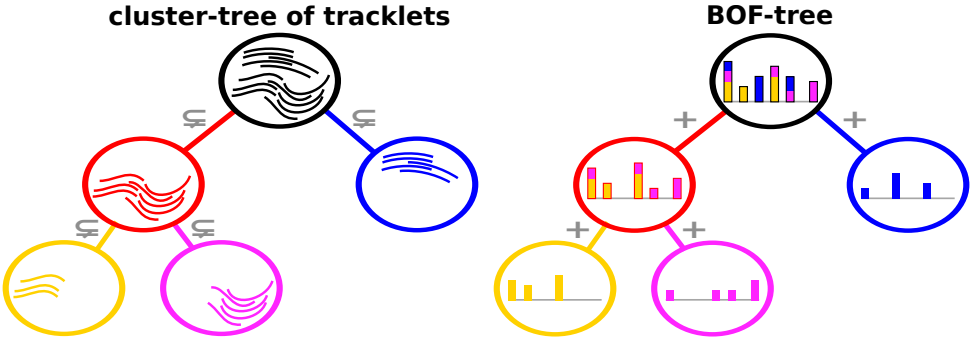

Figure 2: A cluster-tree — where edges between nodes represent a strict inclusion — and its corresponding BOF-tree — where the BOF of a node is the sum of its children's BOFs.

3.3 Kernel between BOF-trees

Inspired by string kernels, existing kernels on variable-size trees are based on measuring structural similarity by counting the number of common sub-structures [43]. In BOF-trees, however, these approaches are not directly applicable as siblings are unordered. In addition, our goal here is to use the structure information to disambiguate comparisons between the *content* of the motion component hierarchies.

We propose the "All Tree Edge Pairs" (ATEP) kernel, which consists in comparing the *edges* of two BOF-trees. Let $\mathcal{T}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{T}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be two BOF-trees, defined from their set of vertices (nodes) $\mathcal{V}_i$ and directed edges (parent-child relations) $\mathcal{E}_i$. Each node $v \in \mathcal{V}_i$ is represented by a BOF — noted $b[v]$ — over its constitutive tracklets. We model a directed edge $e = (v_p, v_c) \in \mathcal{E}_i$ by the *concatenation* — noted $b[e] = (b[v_p], b[v_c])$ — of the BOF of the child node v_c with the BOF of its parent node v_p. Let h be a kernel between BOF — we use the intersection kernel [27] between L_1-normalized histograms. Let $r_1 \in \mathcal{V}_1$ be the root of $\mathcal{T}_1$ and $r_2 \in \mathcal{V}_2$ be the root of $\mathcal{T}_2$. Our ATEP kernel is defined as:

$$k(\mathcal{T}_1, \mathcal{T}_2) = w_r \cdot h(b[r_1], b[r_2]) + \frac{1 - w_r}{|\mathcal{E}_1||\mathcal{E}_2|} \cdot \sum_{\substack{e_1 \in \mathcal{E}_1 \\ e_2 \in \mathcal{E}_2}} h(b[e_1], b[e_2])$$

As the roots have no parents, they are handled separately in this kernel: $w_r \in (0,1)$ is a cross-validated parameter encoding a prior on the importance of the root-to-root comparisons. This kernel relies only on hierarchical relations: a node only depends on its parent. It can be seen as a weighted similarity between all sub-trees of two BOF-trees. Let a *direct family* $(v, s(v), p(v))$ denote, respectively, a non-root node $v \in \mathcal{V}_i \backslash r_i$, its only sibling $s(v)$, and its parent $p(v)$. The additive property of BOF-trees is formulated as $b[v] + b[s(v)] = b[p(v)]$. Therefore, $(b[v], b[p(v)])$ completely characterizes a direct family. In addition, the BOF $b[v]$ is the sum of all the BOFs of its descendants. Consequently, $(b[v], b[p(v)])$ can be seen as an approximation of the content of the sub-tree rooted at $p(v)$. Therefore, the ATEP kernel efficiently compares all sub-trees by using only one level of hierarchy at a time to compare two motion components. Note, that if the node kernel h is positive definite, then our ATEP kernel is also positive definite (it is a sum of positive definite kernels). We can, therefore, use it directly in conjunction with a Support Vector Machine (SVM) [41] classifier.

4 Experiments

4.1 Datasets

Our goal is to show that our hierarchical method allows for an accurate modelling of activity categories that are more complex than traditionally explored actions like running and walking. We, therefore, evaluate the performance of our method on two publicly available recognition benchmarks [34, 36] focusing on two different types of complex activities coming from real-world video sources.

The Olympic Sports dataset [34] contains 783 Youtube videos of athletes practicing 16 different sport activities such as springboard diving and weight lifting. This dataset contains long video sequences of fast and complex articulated human motions, possibly involving interactions with objects (*e.g.* a javelin). It presents several challenges, such as cluttered backgrounds, low quality videos, compression artifacts, and subtle distinctions between some categories (*e.g.* triple jump and long jump). In addition, the activities are composed of multiple simpler actions (*e.g.* running and jumping) that can be shared across categories, which is a great challenge for part-based recognition and justifies the need to leverage the spatio-temporal structure of activities. This dataset, however, has a significant camera motion bias: activities are filmed in a manner that is strongly correlated with the action performed. For instance, diving is consistently shot from below and the camera follows the trajectory of the diver. As our approach relies on dense motion features, we present results on a stabilized version of the Olympics dataset. We follow [19] and found that camera motion stabilization using simple frame stitching techniques leads to good results. In practice, camera stabilization improved results for our method and all baselines described in Section 4.2, because it decreases the confusion between pairs of similar categories filmed in the same manner (*e.g.* the two diving activities). Note, that stabilization is not necessary for our approach.

The High Five dataset [36] consists of 300 video clips collected from 20 different TV shows. The activity categories are four human-human interactions that are difficult to differentiate: hand shakes, high fives, hugs, and kisses. Each activity is performed in 50 different clips, the remaining 100 "negative" clips containing other actions. These activities are of shorter duration and involve a simpler combination of atomic actions compared to the Olympics Sports activities. As the dataset authors propose a model relying on head orientations, annotations for discrete head orientation and upper body localization of actors are available. We do not use this additional supervision in our experiments. In contrast to the Olympics Sports videos, the High Five clips do not suffer from camera motion bias. We, therefore, present results on the original non-stabilized video sequences.

4.2 Baselines

We first compare to two simple baselines using a global BOF model over the entire video [21, 49]. The approach of Laptev *et al.* [21] uses sparse local Spatio-Temporal Interest Points (STIPS) [22] described by a concatenation of histogram of oriented gradients [7] and optical flow, denoted "HOG/HOF". The approach of Wang *et al.* [49] uses dense tracklets represented by MBH descriptors. Note, that this method corresponds to using only the model of the root node of our BOF-trees. In both cases, we use the same vocabulary construction and size as mentioned previously.

In addition to these unstructured baselines, we compare our approach with decompositions obtained by alternative clustering methods. First, we compare to two standard "flat"

SDT trees	**82.7**
SDKM trees	76.7
SDT leaves	77.9
SDKM leaves	72.2
spectral	71.7
kmeans	70.8
Wang *et al.* [49]	75.9
Laptev *et al.* [21]	61.3
Brendel and Todorovic [4]	77.3
Niebles *et al.* [34]	72.1

(a) Olympics Sports (Accuracy in %)

SDT trees	**55.6**
SDKM trees	53.8
SDT leaves	49.5
SDKM leaves	48.6
spectral	48.9
kmeans	50.1
Wang *et al.* [49]	53.4
Laptev *et al.* [21]	36.9
Patron-Perez *et al.* [36]	32.8

(b) High Five (AP in %)

Table 1: Performance comparison on the Olympics Sports [34] and High Five [36] datasets. Results with our Spectral Divisive Thresholding algorithm are noted "SDT". We compare with the state of the art, bag-of-features baselines [21, 49], "flat" decompositions obtained by k-means and spectral clustering, as well as leaves and trees of hierarchical decompositions obtained by a baseline spectral divisive bi-partitioning k-means algorithm noted "SDKM".

clustering algorithms that only produce a set of unrelated clusters: k-means and spectral clustering. Due to the large amount of tracklets per video, we adopt an efficient on-line variant of k-means [42]. For spectral clustering, we adopt the efficient approach of Fowlkes *et al.* [13]: we partition tracklets with (on-line) k-means on the same approximate spectral embedding used by our method. These two algorithms require the desired number of clusters. As this depends on each video, we use a number of clusters that depends linearly on the number of tracklets, such that the smallest videos have at least 2 clusters, whereas the largest ones have at most 1000 clusters. To compare the resulting unstructured sets of clusters, we use a natural simplification of our ATEP kernel consisting in averaging all pairwise cluster comparisons.

We also compare our approach to a closely related baseline yielding a cluster-tree. Its steps are all similar to our method except for one: we replace the spectral divisive thresholding algorithm of Section 2.3 (noted SDT) by a recursive application of k-means. This amounts to splitting a node with (on-line) k-means ($k = 2$) on the spectral embedding. The algorithm is noted "SDKM" for Spectral Divisive K-Means. Like with our approach, we classify the resulting cluster-trees using a SVM with our BOF-tree kernel.

4.3 Results

Table 1 reports performance comparisons between the baselines, the state of the art and our method. Our hierarchical ATEP kernel on SDT BOF-trees in conjunction with a SVM improves over the unstructured BOF baselines. This confirms the importance of leveraging structure information to recognize complex activities. Note, however, that only our method yields clear performance improvements, whereas other structured baselines are less accurate. This shows that properly decomposing activities is a challenging problem that is critical for performance. In particular, using hierarchical relations between motion components with our ATEP kernel consistently improves over the "flat" baselines relying on unrelated sets of clusters such as the leaves of cluster-trees or clusters obtained by k-means. Table 1 also shows that the BOF-trees produced by our SDT algorithm yield more powerful models than the SDKM ones obtained by bi-partitioning k-means. Finally, our approach outperforms the state of the art on both datasets, including latent part models [34] (+10.6%), complex graphical models resulting from video segmentation [4] (+5.4%), and interaction-specific structured learning [36] (+22.8%).

5 Conclusion

In this paper, we introduced an efficient divisive clustering algorithm that hierarchically groups short duration point trajectories. A video is structured as a tree of nested motion components. Each one of these data-driven parts is represented by a bag-of-features over local motion descriptors. We also proposed a new kernel on unordered binary trees, which can accurately compare activities with a variable number of hierarchically ordered parts. This kernel uses an additive property of our motion decomposition to efficiently compare subtrees and leverages structure information to refine part comparisons. Experimental results show that the combination of our clustering algorithm and our tree kernel outperforms unstructured bag-of-features baselines, video decompositions with other clustering techniques, and the state of the art — including approaches based on latent parts, video segmentation, and structured learning.

Acknowledgments This work was partially funded by the MSR/INRIA joint project, the European integrated project AXES and the PASCAL 2 Network of Excellence.

References

[1] S. Ali, A. Basharat, and M. Shah. Chaotic invariants for human action recognition. In *ICCV*, 2007.

[2] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008.

[3] M. Bregonzio, S. Gong, and T. Xiang. Recognising action as clouds of space-time interest points. In *CVPR*, 2009.

[4] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011.

[5] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010.

[6] L.W. Campbell and A.F. Bobick. Recognition of human body motion using phase space constraints. In *ICCV*, 1995.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[8] N. Dalal, B. Triggs, and C. Schmid. Human Detection Using Oriented Histograms of Flow and Appearance. In *ECCV*, 2006.

[9] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, pages 65–72, 2005.

[10] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification and Scene Analysis*. 1995.

[11] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. *Image Analysis*, pages 363–370, 2003.

[12] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2010.

[13] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nystrom method. *PAMI*, 26(2):214–225, 2004.

[14] M. Fradet, P. Robert, and P. Pérez. Clustering point trajectories with various life-spans. In *CVMP*, 2009.

[15] A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011.

[16] A. Gaidon, Z. Harchaoui, and C. Schmid. A time series kernel for action recognition. In *BMVC*, 2011.

[17] M. Grundmann, F. Meier, and I. Essa. 3D shape context and distance transform for action recognition. In *ICPR*, 2008.

[18] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning (2nd edition)*. Springer, 2008.

[19] N. Ikizler-Cinbis and S. Sclaroff. Object, scene and actions: Combining multiple features for human action recognition. In *ECCV*, 2010.

[20] I.N. Junejo, E. Dexter, I. Laptev, and P. Pérez. View-independent action recognition from temporal self-similarities. *PAMI*, 2010.

[21] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[22] Ivan Laptev. On space-time interest points. *IJCV*, 64(2–3):107–123, 2005.

[23] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011.

[24] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *CVPR*, 2009.

[25] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *CVPR*, 2011.

[26] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[27] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008.

[28] M. Marszalek, I. Laptev, and C. Schmid. Actions in contexts. In *CVPR*, 2009.

[29] P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 514–521. IEEE, 2009.

[30] P. Matikainen, M. Hebert, and R. Sukthankar. Representing pairwise spatial and temporal relations for action recognition. *ECCV*, 2010.

[31] É. Mémin and P. Pérez. Joint estimation-segmentation of optic flow. In *ECCV*, 1998.

[32] R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV*, 2009.

[33] J.C. Niebles and L. Fei-Fei. Hierarchical model of shape and appearance for human action classification. In *CVPR*, 2007.

[34] J.C. Niebles, CW. Chen, , and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010.

[35] V. Parameswaran and R. Chellappa. View invariance for human action recognition. *International Journal of Computer Vision*, 66(1):83–101, 2006.

[36] A. Patron-Perez, M. Marszałek, A. Zisserman, and I. D. Reid. High five: Recognising human interactions in TV shows. In *BMVC*, 2010.

[37] A. Prest, V. Ferrari, and C. Schmid. Explicit modeling of human-object interactions in realistic videos. Technical report, INRIA, 2011. URL http://hal.inria.fr/inria-00626929.

[38] C. Rao, A. Yilmaz, and M. Shah. View-invariant representation and recognition of actions. *International Journal of Computer Vision*, 50(2):203–226, 2002.

[39] M. Raptis, I. Kokkinos, and S. Soatto. Discovering discriminative action parts from mid-level video representations. In *CVPR*, 2012.

[40] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *IJCV*, 80(1):72–91, 2008.

[41] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[42] D. Sculley. Web-scale k-means clustering. In *WWW*, 2010.

[43] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge Univ Pr, 2004.

[44] Y. Sheikh, M. Sheikh, and M. Shah. Exploring the space of a human action. In *ICCV*, 2005.

[45] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *ICCV*, pages 1154–1160. IEEE, 1998.

[46] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[47] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *PAMI*, 26(4):479–494, 2004.

[48] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.

[49] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action recognition by dense trajectories. In *CVPR*, 2011.

[50] Y. Wang and G. Mori. Hidden part models for human action recognition: Probabilistic vs. max-margin. *PAMI*, 2011.

[51] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008.

[52] CW Zitnick, N. Jojic, and S.B. Kang. Consistent segmentation for optical flow estimation. In *ICCV*, 2005.