Space-Time Tree Ensemble for Action Recognition

Shugao Ma
Boston University
shugaoma@bu.edu

Leonid Sigal
Disney Research
lsigal@disneyresearch.com

Stan Sclaroff
Boston University
sclaroff@bu.edu

Abstract

Human actions are, inherently, structured patterns of body movements. We explore ensembles of hierarchical spatio-temporal trees, discovered directly from training data, to model these structures for action recognition. The hierarchical spatio-temporal trees provide a robust mid-level representation for actions. However, discovery of frequent and discriminative tree structures is challenging due to the exponential search space, particularly if one allows partial matching. We address this by first building a concise action vocabulary via discriminative clustering. Using the action vocabulary we then utilize tree mining with subsequent tree clustering and ranking to select a compact set of highly discriminative tree patterns. We show that these tree patterns, alone, or in combination with shorter patterns (action words and pairwise patterns) achieve state-of-the-art performance on two challenging datasets: UCF Sports and HighFive. Moreover, trees learned on HighFive are used in recognizing two action classes in a different dataset, Hollywood3D, demonstrating the potential for cross-dataset generality of the trees our approach discovers.

1. Introduction

Automatic recognition of human actions in video is important for applications in surveillance, HCI, video search and retrieval. Human actions, or interactions, are inherently defined by structured patterns of the humans' movement. As such, human actions can be modeled as spatio-temporal graphs, where the graph vertices encode movements of whole body or body parts, and the graph edges encode spatio-temporal relationships between pairs of movement elements, for instance temporal progression, *e.g.*, one movement followed by another movement, spatial composition, *e.g.*, movement of upper body coupled with movement of lower extremities, or even hierarchical relationships of elements, *e.g.*, movement of the body as a whole can decompose into local movements of the limbs.

A single spatio-temporal structure, however, is unlikely to be sufficient to represent a class of action in all but the simplest scenarios. First, the execution of the action may differ from subject to subject, involving different body parts or different space-time progressions of body part movements. Second, the video capture process introduces intra-class variations due to occlusions or variations in camera viewpoint. Thus, the resulting space-time and appearance variations necessitate using a *collection* of spatio-temporal structures that can best represent the action at large.

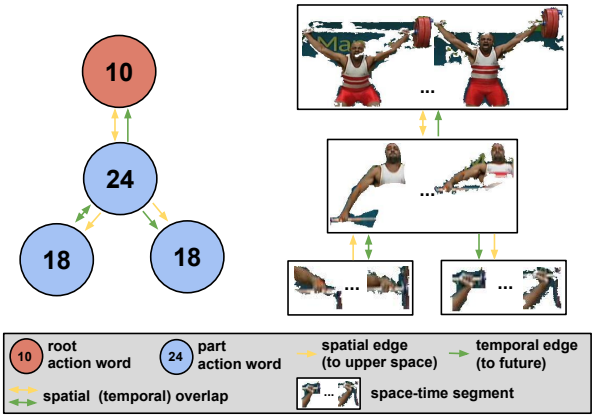

Figure 1. One example tree structure discovered by our approach for the *lifting* action and its best match in a testing video. In the tree, one node (red) indexes to a root action word and is matched to an STS of the upward movement of the whole body; three nodes (blue) index to the part action words and are matched to STSs of the upward movement of the upper-body and two temporally consecutive movements of the left arm and lower left arm respectively.

In this work, we propose a formulation that discovers a collection of hierarchical space-time trees from video training data, and then learns a discriminative action model that builds on these discovered trees to classify actions in videos. Fig. 1 illustrates one simple discovered tree and its best match in a testing video and Fig. 2 shows more examples. Given a video, we first extract a collection of space-time segments (STSs); each STS is a sub-volume that can be produced by video segmentation, and it may cover the whole human body or a body part in space-time. We explore the hierarchical, spatial and temporal relationships among the STSs; this transforms a video into a graph. We

discover a compact set of frequent and discriminative tree structures from graphs of training videos and learn discriminative weights for the trees' nodes and edges. Finally, we construct action classifiers given the detection responses of these trees.

We use trees instead of graphs for multiple reasons: first, any graph can be approximated by a set of spanning trees; second, inference with trees is both efficient and exact; third, trees provide a compact representation as it is easy to account for multiple structures using a single tree by allowing partial matching during inference (which is no more expensive). Partial matching of trees during inference also allows us to effectively deal with variations in action performance and segmentation errors.

We note that, as the size of the tree structures increases the trees get more specific, thereby capturing more structural information that can provide greater discriminative power in classification. On the other end of the spectrum, smaller structures, particularly singletons and pairs, tend to appear more frequently in their exact forms than larger tree structures. Thus, smaller structures can be helpful when the action is simple but the occlusions or video segmentation errors are significant. Therefore, for robust action classification, the best strategy is to flexibly exploit both the large and small structures in a unified model.

Contributions: Our contributions are as follows: (1) We propose an approach that enables discovery of rich high-level tree structures that capture space, time and hierarchical relationships among the action words. (2) We propose a discriminative action model that utilizes both small structures (*i.e.* words and pairs) and richer, tree structures; this unified formulation achieves state-of-the-art performance in recognizing and localizing human actions and interactions in realistic benchmark video datasets. (3) We show generalization of the learned trees by cross-dataset validation, achieving promising results on the Hollywood3D dataset using trees learned on the HighFive dataset.

2. Related Work

Bag-of-Words (BoW) representations can be quite effective for action recognition. Space-time features are often computed from space-time interest point operators [14, 37], point trajectories [28, 29, 12] or video segments [15]. Usually a large codebook, with thousands of words, is built by clustering these local features. The videos are then represented as (normalized) histogram counts of local space-time features over the codebook, on which action classifiers, *e.g.* SVMs, are trained. While effective in practice, these representations lack the spatio-temporal structure necessary to model progression or structure of an action.

One way to encode spatio-temporal structures is to impose a fixed spatio-temporal grid over the video and construct a BoW representation by concatenating the BoW representations from each grid cell [14, 25, 18]. While this encodes spatio-temporal structure, the fixed spatio-temporal grid only offers very coarse structural information and typically does not align well with the spatial location or temporal progression of the action. Other works have tried to encode simple structures more explicitly [16, 36]. In [16], quantized local features are augmented with relative space-time relationships between pairs of features; in [36] a visual location vocabulary is computed to incorporate spatio-temporal information.

Richer structures have also been explored. Generative models such as HMMs have been used, *e.g.*, [21, 34, 11]; however, the independence assumptions of HMMs are often not held in practice. Discriminative models, *e.g.*, HCRF, are also widely used [32, 33, 22, 23]. In each of these works a single manually defined structure is used to model human action as a constellation or temporal chain of action parts. In contrast, we model an action as an ensemble of tree structures. More importantly, we discover the structures from data as opposed to defining them by hand. [30] learns mixture models of body parts and spatio-temporal tree structures using annotated human joints on video frames. Our method only requires action labels of the videos.

Subgraph mining has also been used for action recognition [1]. However, subgraph mining techniques only consider exact matching of subgraphs and discover graphs that are frequent, not necessarily discriminative. We use subgraph mining only to produce a large set of candidate subtrees, from which, using the proposed clustering and ranking methods, we discover a compact subset of discriminative and non-redundant subtrees.

Other approaches transform videos into graphs and classify actions based on these graphs. In contrast to these methods, which attempt to learn a single holistic graph per action class [2, 27] or a graph kernel that measures compatibility between whole graphs [31, 7, 35], we focus on identifying frequent and discriminative subtrees. This allows for a more versatile and compact representation that we also observe is often semantically meaningful.

3. Model Formulation

To capture both the appearance of local space-time segments (STSs) and the hierarchical, spatial and temporal structures among them, we transform a video into a graph where the nodes are STSs and the edges are labeled with the hierarchical, spatial and temporal relationships among the STSs. The graph nodes are subsequently given labels that are indices to a compact but discriminative action word vocabulary. Thus, in training, the training video set is converted to a set of labeled graphs, from which we discover a collection of discriminative tree structures for each action.

Formally, a video is represented as a graph $G =$

$\{V,\ A^t,\ A^s,\ A^h,\ F\}$. V is the set of vertices that are the STSs. A^t, A^s and A^h are the time, space and hierarchical adjacency matrices containing edge labels (details in Sec. 5). The rows of matrix $F = [f_1, f_2, ..., f_{|V|}] \in \mathbb{R}^{|V|\times d}$ are visual features (*e.g.* HoG features) extracted from the STSs. For each action class a, a collection of trees is then used in constructing an ensemble classifier:

$$S_a(G, \mathcal{T}) = \mathbf{w}^T \cdot \Phi(G, \mathcal{T}) = \sum_{m\in\{1,...,|\mathcal{T}|\}} w_m \phi_m(G, \mathcal{T}_m), \tag{1}$$

where G denotes a test input video, $\mathcal{T}$ is the set of learned tree structures for class a and $\mathcal{T}_m$ is one of such trees in this set, and $\mathbf{w} = \{w_m; m \in \{1, ..., |\mathcal{T}|\}\}$ is the learned weight vector. Each ϕ_m is a scoring function that measures compatibility (or degree of presence) of $\mathcal{T}_m$ in video G. [1] In the multi-class classification setting, the predicted action class a^* of G is computed by $a^* = arg\max_a\ S_a(G, \mathcal{T})$.

We formalize a tree as $\mathcal{T}_m = \{N,\ E^t,\ E^s,\ E^h,\ \boldsymbol{\beta}\}$ where N, $\{E^t, E^s, E^h\}$ are the nodes and adjacency matrices respectively. $\boldsymbol{\beta}$ are discriminative weights associated with the nodes and edges. Each node $n_i \in N$ is an index into a learned discriminative action word vocabulary $\mathcal{W}_a$ for class a (described in Sec. 6); each edge E^k_{ij} ($k \in \{t, s, h\}$) is associated with a corresponding temporal, spatial or hierarchical relationship between nodes i and j, similar to the relations defined for A^k in graph G. The matching score of a tree to a graph is computed as follows:

$$\phi_m(G,\ \mathcal{T}_m) = \psi\left(\{\ \boldsymbol{\beta}\cdot\varphi(G, \mathcal{T}_m, \mathbf{z})\ |\ \mathbf{z} \in Z(G, \mathcal{T}_m)\}\right), \tag{2}$$

where $\mathbf{z}$ is latent variable that represents a match of a tree $\mathcal{T}_m$ to the video G: $\mathbf{z}$ is realized as $\mathbf{z} = (z_1, ..., z_{|N|})$ where z_i is the index of the vertex in G that is matched to the ith node in $\mathcal{T}_m$. ψ is a pooling function over the matching scores of the set of all possible (partial) matches $Z(G, \mathcal{T}_m)$. The matching score of a specific match $\mathbf{z}$ to $\mathcal{T}_m$ is:

$$\begin{aligned}\boldsymbol{\beta}\cdot\varphi(G, \mathcal{T}_m, \mathbf{z}) = &\sum_{n_i\in N} \beta_i\ p_n(z_i,\ n_i) \\ &+ \sum_{k\in\{t,s,h\}}\ \sum_{\substack{E^k_{ij}\in E^k \\ E^k_{ij}\neq 0}} \beta^k_{ij}\ p_k(A^k_{z_i z_j},\ E^k_{ij}).\end{aligned} \tag{3}$$

where β_i and β^k_{ij} ($k \in \{t, s, h\}$) are the tree node weights and edge weights respectively. The function p_n scores compatibility of the tree nodes with graph vertices; p_t, p_s and p_h score compatibility of the temporal, spatial and hierarchical graph edges with tree edges. Partial matching is possible by adding a *null* vertex $v_\varnothing$ to V as the 0th vertex and also adding a 0th row and column of zeros to A^s, A^t and A^h. Any node in $\mathcal{T}_m$ not matched to a vertex in G is assigned to match the 0th vertex.

Our focus is on *discovering the tree structures* $\mathcal{T}$. These structures, their parameters $\boldsymbol{\beta}$, as well as parameters of the ensemble $\mathbf{w}$ are learned directly from the data. Given a tree structure, parameters can be learned in variety of ways, *e.g.*, using latent SVM [33]. However, discovering the tree structures themselves is the key challenge as: (1) the space of tree structures is exponential in the number of tree nodes and types of relationships allowed among the tree nodes; (2) partial presence of the trees needs to be considered; (3) without annotation of body parts, the tree nodes themselves are to be discovered. Also note that our model unifies small structures, *e.g.* singletons in the extreme case, with rich tree structures: when the trees are singletons, the model essentially reduces to the BoW model.

4. Inference

Given a set of discovered tree structures $\mathcal{T}$, learned parameter vectors $\mathbf{w}, \boldsymbol{\beta}$ and a test video represented by a graph G, the score $S_a(G, \mathcal{T})$ of G containing an action a can be computed as a weighted sum of independently matched tree scores $\phi_m(G, \mathcal{T}_m)$. If we use average pooling in Eq. 2, then

$$\phi_m(G, \mathcal{T}_m) = \frac{\sum_{\mathbf{z}\in Z(G,\mathcal{T}_m)} \boldsymbol{\beta}\ \cdot\ \varphi(G, \mathcal{T}_m, \mathbf{z})}{|Z(G, \mathcal{T}_m)|}, \tag{4}$$

which requires looping through the whole possible set of latent values $Z(G, \mathcal{T}_m)$, which is expensive for trees with three or more nodes. Max pooling is more appropriate for larger trees:

$$\phi_m(G, \mathcal{T}_m) = \max_{\mathbf{z}\in Z(G,\mathcal{T}_m)} \boldsymbol{\beta}\ \cdot\ \varphi(G, \mathcal{T}_m, \mathbf{z}). \tag{5}$$

This can be efficiently computed by dynamic programming (DP), which we describe next.

Recall that every tree node n_i is an index into $\mathcal{W}_a$ and every z_i is an index into video graph vertices that are associated with features, we define the potentials in Eq. 3 as:

$$p_n(z_i,\ n_i) = \begin{cases} e^{c_{n_i}(f_{z_i})}, & c_{n_i}(f_{z_i}) \geq \delta \\ 0, & otherwise \end{cases} \tag{6}$$

$$p_k(A^k_{z_i z_j},\ E^k_{ij}) = \begin{cases} 1, & A^k_{z_i z_j} = E^k_{ij} \\ 0, & otherwise \end{cases} \tag{7}$$

where $k \in \{t, s, h\}$. The set of classifiers $c_{n_i}(f_{z_i}) \in \mathbb{R}$ score the mapping of the z_ith graph vertex (associated with feature f_{z_i}) to the action word which tree node n_i indexes to. The reader is referred to [5] for the details of DP on a tree, but note when matching n_i, the set of possible choices for graph vertices can be pruned for efficiency to those for which $c_{n_i}(f_{z_i}) \geq \delta$; further, nodes that cannot be matched are assigned to $v_\varnothing$ to allow partial tree matches.

[1] To avoid notation clutter, we omit the action class label a for $\mathcal{T}$, $\mathbf{w}$, Φ, ϕ and φ.

The DP procedure may match a graph vertex (STS) to multiple tree nodes if those tree nodes index to the same action word. It is possible to alleviate this, *e.g.*, as in [22], but this would in general introduce high-order potentials requiring more expensive or approximate inference. We find that assigning multiple nodes is not problematic and can be beneficial in practice to account for video segmentation errors.

5. Discovering Structures

In Sec. 3 we describe how a video is represented as a graph of STSs. STSs are space-time sub-volumes of the video and can be produced by video segmentation methods. We use the method in [15] to extract STSs, because such STSs are shown to be effective for action recognition and localization. Given a video, [15] extracts two types of STSs that can include both static and non-static body parts: root STSs often covering whole human body and part STSs often covering body parts. Note no person detector nor human bounding box annotations are required for this procedure.

We use a small discrete set for edge labels: $A^t \in \{0, \leftarrow, \rightarrow, \bowtie\}^{|V|\times|V|}$, where $\leftarrow, \rightarrow$ and $\bowtie$ denote *after*, *before* and *temporal overlap*; similarly $A^s \in \{0, \uparrow, \downarrow, \bowtie\}^{|V|\times|V|}$, where $\uparrow, \downarrow$ and $\bowtie$ denote *above*, *below* and *spatial overlap*; $A^h \in \{0, r \rightarrow r, r \rightarrow p, p \rightarrow r, p \rightarrow p\}^{|V|\times|V|}$ denotes heirarchical root (r) - part (p) relationships. Using coarse discrete labels helps make the representation more robust to variations in action execution and STS segmentation errors.

Specifically, for a pair of the ith and jth STS in a video: (1) A^h_{ij} is set according to the part / root identity of i and j; (2) A^t_{ij} and A^s_{ij} are set as:

$$A^t_{ij} = \begin{cases} \bowtie, & |t_i - t_j| \leq \delta_t \\ \leftarrow, & t_i - t_j > \delta_t \\ \rightarrow, & t_j - t_i > \delta_t \end{cases} \tag{8}$$

$$A^s_{ij} = \begin{cases} \bowtie, & |y_i - y_j| \leq \delta_s \\ \uparrow, & y_j - y_i > \delta_s \\ \downarrow, & y_i - y_j > \delta_s \end{cases} \tag{9}$$

where t_i and t_j are the starting frame indices of i and j, y_i and y_j are the mean vertical center positions of i and j over the frames in which i and j co-exist; δ_t and δ_s are constants fixed to 3 (frames) and 20 (pixels) in all our experiments. If the temporal spans of i and j are disjoint, we let $A^s_{ij} = 0$.

Instead of constructing a complete graph, which can lead to high computational cost in the subsequent tree mining procedure, we construct a sparse graph. We group the STSs into tracks by the method in [15], and only add edges for: 1) each pair of root STSs; 2) each pair of root STS and part STS in the same track; 3) each pair of part STSs in the same track that have significant temporal overlap.

5.1. Discovering Tree Structures

We want to find frequent and discriminative tree structures for action classification (Fig. 2). Enumerating the set of all possible trees is infeasible due to its exponential size. Furthermore, in practice, it is hard to find exact presence of complex trees due to noise in the STS extraction and variation of action execution, so we must account for partial matches when counting frequencies. We propose a two step approach: first, we mine a large and redundant set of trees using a tree mining technique that only counts exact presence; second, we discover a compact set of trees by clustering and ranking the mined trees using a form of cosine distance on their discriminative parameter vectors $\boldsymbol{\beta}$.

Tree Mining: We use the subgraph mining algorithm GASTON [17] to mine frequent trees with at most six vertices. The number of mined trees is controlled by the minimum support threshold parameter for each class so that $5\times10^3 \sim 1\times10^4$ trees are mined per action class. For each action class, we mine trees from both positive videos and negative videos and remove the trees that appear in both.

We then train discriminative parameters $\boldsymbol{\beta}$ for each mined tree. Given a mined tree $\mathcal{T}_m$, we initialize $\boldsymbol{\beta}$ by setting all node weights to 1 and edge weights to 0.5. We find the best matches of the tree in the graphs constructed from training videos, using the inference procedure in Sec. 4. $\boldsymbol{\beta}$ is then refined by training a linear SVM on the set of best matches. This procedure is similar to performing one iteration of latent SVM, except much faster.

Tree Clustering: The trees we mine using GASTON tend to be highly redundant, especially if one considers partial matching. To avoid the redundancy and discover a set of trees that effectively cover intra-class variations, we cluster the mined trees and select the best tree from each cluster. To compute similarity between two trees, $\mathcal{T}_m$ and $\mathcal{T}_{\hat{m}}$, we utilize the inference procedure in Sec. 4 by treating one of the trees (*e.g.*, $\mathcal{T}_{\hat{m}}$) as a weighted graph with weights on the nodes and edges corresponding to learned parameters $\hat{\boldsymbol{\beta}}$. The potentials in $\phi_m(\mathcal{T}_{\hat{m}}, \mathcal{T}_m)$ for Eq. 3 are then altered as follows:

$$p_n(z_i,\ n_i) = \begin{cases} \hat{\beta}_{z_i}, & \hat{n}_{z_i} = n_i \\ 0, & \hat{n}_{z_i} \neq n_i \text{ or } z_i = v_\varnothing \end{cases} \tag{10}$$

$$p_k(\hat{E}_{z_i z_j},\ E_{ij}) = \begin{cases} \hat{\beta}^k_{z_i z_j}, & \hat{E}^k_{z_i z_j} = E^k_{ij} \\ 0, & otherwise \end{cases} \tag{11}$$

where $\hat{n}_{z_i}$ is the z_ith tree node of $\mathcal{T}_{\hat{m}}$ and $k \in \{t,\ s,\ h\}$. Note that this procedure will give us essentially the *edit* cosine distance (cosine distance since when structures are the same, $\phi_t(\mathcal{T}_{\hat{m}}, \mathcal{T}_m) = \boldsymbol{\beta} \cdot \hat{\boldsymbol{\beta}}$). This similarity measure between trees takes into account both similarity in structure and similarity in discriminative parameters $\boldsymbol{\beta}$.

After computing the similarities between each pair of trees, we run affinity propagation [6] to cluster the trees and

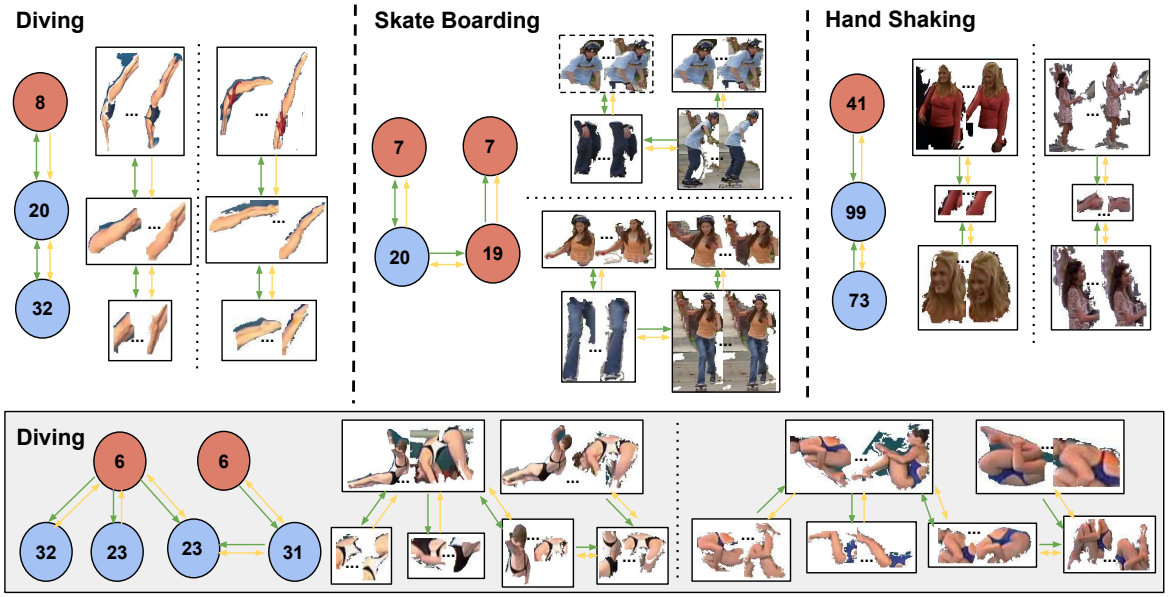

Figure 2. **Examples of discovered trees.** For each tree, we also show inference results on two testing videos. See the legend of Fig 1 for the meaning of the figure components. The space-time segment enclosed by dotted box is matched to multiple tree nodes.

pick the tree that has the highest cross validation accuracy in each cluster as the non-redundant prototype.

Tree Ranking: The above procedure still tends to find a relatively large number of trees (≈ 150 per class or more). Our hypothesis, which we validate in experiments, is that only a subset is needed for classification. To pick the best candidates we need some measure of quality among the candidates. We find cross-validation to be unstable here, and instead propose ranking based on entropy. We compute an *activation entropy* for each selected tree $\mathcal{T}_m$:

$$Entropy(\mathcal{T}_m, \mathcal{G}) = -\sum_c P_c(\mathcal{T}_m, \mathcal{G})\ log(P_c(\mathcal{T}_m, \mathcal{G})), \tag{12}$$

$$P_c(T, \mathcal{G}) = \frac{|\{G \mid \forall\, G \in \mathcal{G},\ \phi_m(G, \mathcal{T}_m) \geq 0,\ \mathcal{L}(G) = c\}|}{|\{G \mid \forall\, G \in \mathcal{G},\ \phi_m(G, \mathcal{T}_m) \geq 0\}|}, \tag{13}$$

where $\mathcal{G}$ is the training set and $\mathcal{L}(G)$ is class label of G. The numerator encodes the number of videos in class c that are classified as positive using $\mathcal{T}_m$ and denominator the number of videos classified as positive over the training set. Intuitively, trees with smaller activation entropies have more consistent patterns in classification errors, and are more likely to be stable structures in human actions.

Despite the large set of trees that we mine, in the experiments we illustrate that by using only a compact set of trees attained through the above process (20 trees per action in UCF-Sports, and 50 per action in HighFive) state-of-the-art recognition performance is achieved.

5.2. Discovering Pairwise Structures

When occlusion or video segmentation errors are significant, it is easier to find exact matches for smaller structures than larger ones. Pairs in particular are shown to be effective for classification *e.g.* as in [16]. Our tree discovering approach described in the previous section can find useful pairwise structures, but is by no means exhaustive. Meanwhile, our formulation of a video as a graph enables a simple exhaustive search for a set of compact and yet discriminative pairwise structures.

Specifically, denote $\mathcal{P}$ as the space of pairs for action class a, then $|\mathcal{P}| = (|\mathcal{W}_a| - 1)|\mathcal{W}_a||\mathcal{E}|/2$, where $\mathcal{E}$ denotes the space of edge labels and in our case $|\mathcal{E}| = 3{\times}3{\times}4 = 36$. In our experiments $|\mathcal{W}_a|$ is $50 \sim 150$, so $|\mathcal{P}| \leq 4 \times 10^5$. We construct a histogram $H \in \mathbb{R}^{|\mathcal{P}|}$ for each training graph G. The pth bin, $H_p = \kappa_p / (\sum_{i \in \{1,\ldots,|\mathcal{P}|\}} \kappa_i)$, where κ_p denotes appearance counts of the pth pair in G. We then compute the mean H_p^+ over the graphs of positive training videos. Recall that we are after frequent and discriminative structures. Since our action words are discriminatively trained, our main criterion for selecting a pair is frequency H_p^+. As allowing partial matching for pairs may reduce to matching action words, we require exact matching in inference and set node weights $\beta_1 = \beta_2 = 1/2$, edge weights $\boldsymbol{\beta}_{12}^k = 0$ and $p_k(A_{z_1 z_2}^k,\ E_{12}^k) = 1$ for $k \in \{t, s, h\}$.

6. Building the Action Word Vocabulary

In BoW approaches, large vocabularies with several thousands of words are typically used. We argue that if

Figure 3. **Representative action words:** for each action word, we show 5 space-time segments (illustrated by one temporal slice) that are in the cluster corresponding to the action word.

the words are discriminative enough, a small vocabulary can perform at least equally well as larger ones. Here we propose a method to construct a compact and yet discriminative action word vocabulary, which can be one order of magnitude smaller than the one used in [15] while achieving similar performance in our experiments. Furthermore, such small vocabularies help greatly reduce the complexity of tree discovery, as the space of trees of length m is exponential in the size of this vocabulary, *i.e.*, $\mathcal{O}(|\mathcal{W}|^m)$. Fig. 3 shows some representative action words. We see that the STS clusters for these action words tend to be coherent and semantically meaningful.

Specifically, we learn the set of action words by discriminative clustering of the STSs. Recall there are two types of STSs extracted: root STSs and part STSs. Clustering is done separately for roots and parts; as a result, we get clusters $\mathcal{W}_i^r$ and $\mathcal{W}_i^p$ for the roots and the parts respectively and $\mathcal{W}_i = \mathcal{W}_i^r \cup \mathcal{W}_i^p$. We use discriminative subcategorization (DSC) [10] to perform the clustering in each action class i, where the negative samples are the STSs extracted from videos of other action categories. DSC treats the cluster memberships of samples as latent variables and jointly learns one linear classifier per cluster using latent SVM. As we believe the vocabulary should be data-driven and adapt to the complexity of the dataset and action class, we perform an initial clustering using the affinity propagation algorithm [6] to decide the number of clusters and the initial cluster membership for each sample.

7. Learning the Ensemble

The main focus of our work is on discovery of larger space-time tree structures (Sec. 5.1). We also consider pairwise structures (Sec. 5.2) and trivial structures of action words (Sec. 6) as special cases that can be discovered through exhaustive search or clustering. We observe that larger structures (*i.e.*, trees) tend to be more discriminative than smaller ones (*i.e.*, action words and pairs), but appear in their exact form less frequently (thus, inexact matching is crucial). In practice, we have found that a combination of simpler and more complex structures tends to offer more robustness. Given the set of trees, and their responses on the training set obtained by inference on the training set of videos, we train parameters $\mathbf{w}$ using linear SVM jointly (using multi-class SVM [3]). When combining words, pairs and trees, we experiment with both early fusion and late fusion. In early fusion, the inference responses of different types of structures are concatenated and the ensemble weights are learned using SVM. In late fusion, one SVM is trained per structure type and we then train a separate SVM to combine their outputs.

8. Experiments

Setup: We test our methods on two benchmark datasets: UCF-Sports [24] and HighFive [20]. *UCF-Sports* contains 150 videos of 10 different sports actions. We use the training / testing split provided by [13], in which 47 videos are used for testing and the rest for training. *HighFive* contains 300 videos from TV programs. 200 of the videos contain 4 different interactions and the other 100 are labeled as negative. We use the training / testing split that is provided in the dataset. Both datasets provide bounding box annotations, which we do not use in training. To ease comparison with results reported in previous works, action recognition performance is measured in terms of mean average precision on HighFive and mean per-class accuracies on UCF-Sports.

Implementation: We extract hierarchical space-time segments from videos using the code provided by [15] with the same parameters. From each space time segment, the same types of raw features as [15] are extracted: HoG, HoF and MBH on UCF-Sports and MBH on HighFive. We empirically found that max pooling works better for action words and trees, and average pooling is better for pairs. Thus, when combining them (Table 3), we use max pooling for words and trees and average pooling for pairs. We learn the ensemble parameters by using liblinear [4].

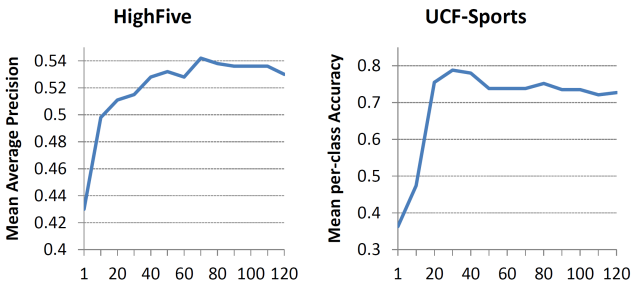

Figure 4. Action recognition performance vs. number of trees. The performance improves quickly when more than one tree is used, but such gain becomes small after approximately the first 20 trees.

Action Word Vocabulary: The total number of action words we discover is approximately 700 for HighFive and 600 for UCF-Sports. This is an order of magnitude smaller than in [15], but similar performance is achieved using only these action words (see Table 3).

Quality of Tree Discovery: Our goal is to discover a compact set of frequent and discriminative structures that can be used to attain good action recognition performance. Therefore, one critical question is how many structures should be selected from our discovered and ranked list of trees (ranked by $Entropy(\mathcal{T}_m, \mathcal{G})$) and pairs (ranked by H_p^+). Fig. 4 illustrates the performance as a function of the selected trees. We note that with a single tree, the performance is low; this indicates the need for an ensemble. However, performance improves quickly, which validates our ranking. After the first $\sim$ 20 trees, the performance gains are small. For pairs, the trend is similar but significantly more pairs are need than trees to achieve the same level of performance. Based on these observations, we choose to use 50 trees and 100 pairs per class for HighFive, and 20 trees and 40 pairs per class for UCF-Sports.

Action Classification: Comparison of our full model to the state-of-the-art methods is given in Tables 1 and 2. Since we use the same space-time segments and low level features as [15], the gains of the hierarchical, spatial and temporal structures we discover are best illustrated in the comparison with their bag-of-words approach: we outperform by more than 8% on both datasets. We also compare with other methods that use different video features. In contrast to our simple linear model, [7, 31] rely on complex graph kernels. [26, 22, 13] model human actions with space-time structures, but the number of nodes in the structures and their inter-connections are pre-determined; moreover, in training they require human bounding box annotations on video frames. Note our method automatically discovers the structures and only needs action labels of the training videos. Nonetheless, our approach consistently achieves better performance, improving on the state-of-the-art.

Analysis of Different Components: We decompose our full model and analyze the performance impact of each component, *i.e.* the action words, pairs and trees. The results of this analysis are reported in Table 3. Several observations can be made. *First*, using only trees, pairs or action words, we attain performance comparable to state-of-the-art. *Second*, although the total number of action words is an order of magnitude smaller than the codebooks used in [15, 28], similar performance is achieved using only these action words. Our compact set of action words greatly reduces the search space for tree structure discovery, without sacrificing discriminative power. *Third*, in all cases, combination works better than using a single type, especially when combining pair and tree structures with action words. This shows that the words, pairs and trees are complementary. We posit that larger structures are more discriminative but less frequently appear in their exact forms than smaller ones, leading to the complementarity observed.

Tree Size: We analyze the impact of tree size on the action classification performance for UCF-Sports in Fig 5, where the tree size is measured as the number of tree nodes. For trees of a given size, we order the trees by their *activation entropy* (Eq. 12) and measure the action classification performance when varying the number of trees used. When using only one tree, larger trees tend to outperform smaller trees. When more trees are used, larger trees produce better performance than smaller trees for most of the cases. These results show the increased discriminative power as

Method	mAP
Our (*Early Fusion* **w+t+p**)	62.7
Our (*Late Fusion* **w+t+p**)	**64.4**
Gaidon *et al.*[7]	62.4
Ma *et al.*[15]	53.3
Laptev *et al.*[14]	36.9
Patron-Perez *et al.*[19]	42.4

Table 1. Mean average precision (mAP) on the HighFive dataset.

Method	Accuracy
Ours (*Early Fusion* **w+t+p**)	**89.4**
Ours (*Late Fusion* **w+t+p**)	86.9
Wang *et al.*[31]	85.2
Ma *et al.*[15]	81.7
Raptis *et al.*[22]	79.4
Tian *et al.*[26]	75.2
Lan *et al.*[13]	73.1

Table 2. Mean per-class accuracy on UCF Sports dataset.

	w	**p**	**t**	**w+p**	**w+t**	**p+t**	**w+p+t**
High5	53.5	48.9	52.4	60.1	57.7	57.6	**62.7**
UCF	78.8	73.4	75.5	85.2	83.6	80.2	**89.4**

Table 3. Action recognition performance (mean average precision for HighFive and mean per-class accuracy for UCF-Sports) using only words, pairs or trees, as well as their combinations. Early fusion is used for combining different types of structures.

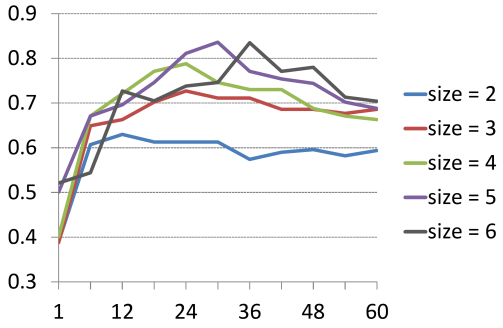

Figure 5. Action classification performance under different tree sizes on the UCF-Sports dataset. Larger trees outperform smaller ones for most of the cases.

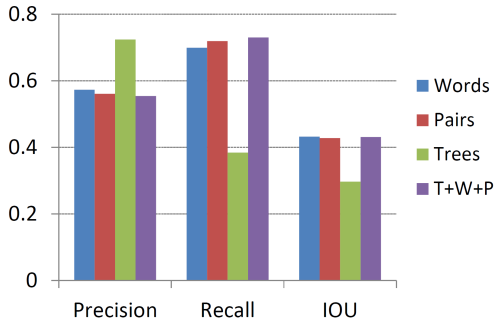

Figure 6. Action localization results on UCF-Sports. Trees may miss some body parts, resulting in lower recall and IOU, but they localize to the action performer(s) with very high precision.

we capture more complex hierarchical, spatial and temporal structures in the human actions. However, we also argue that smaller tree structures are complementary to larger tree structures. For instance, for videos of simple actions but significant occlusion or video segmentation errors, smaller trees can be helpful when used in combination with larger tree structures; the combination attains state-of-the-art performance (Table 3).

Action Localization: Our method can predict the action location by computing the regions of the STSs that have positive contribution in classification. As human bounding box annotations in HighFive only contain head and shoulders, which makes evaluation less meaningful, we only consider UCF-Sports here. Overall, the localization performance is similar to [15] (see Fig. 6), when measured by intersection-over-union (IOU) following [15]. More interestingly, using only trees, the precision (*i.e.*, percentage of predicted area that is within annotated bounding box) is much higher (by $\sim 15\%$) than using only words or pairs, but the recall (*i.e.*, percentage of annotated bounding box areas that are predicted) is lower. Because the trees are learned in a discriminative approach, they may sometimes ignore body parts that are not discriminative; nevertheless, trees can more precisely localize the human(s) engaged in the action.

Method	Kiss	Hug	Avg
Ours (trees from HighFive)	20.8	27.4	24.1
Hadfield *et al.*[8]	10.2	12.1	11.15
Hadfield *et al.*[9]	31.3	32.4	31.9

Table 4. Average Precision on Hollywood3D dataset.We use 50 trees learned for the Kiss and Hug classes on HighFive receptively. Only the left view of the RGB sequences are used, without using any multi-view or depth information that are explored in [8, 9].

8.1. Cross-Dataset Validation

Our approach discoveres trees that capture the discriminative hierarchical and spatial-temporal structures in human actions. These learned representations are generic, and could potentially be effective for recognizing actions in a dataset different from the training dataset. To test this idea, we use the action word vocabulary and trees learned for Kiss and Hug on the HighFive dataset, and train action classifiers as ensembles of trees on Hollywood3D [8]. Note, only the ensemble weights (*i.e.*, w_m in Eq. 1) are re-trained on Hollywood3D. We choose *Kiss* and *Hug* because they are the only action classes that appear in both datasets. We only use the left view of the RGB sequences (no multiview or depth information is explored).

The results are shown in Table 4. Note the compared methods [8, 9] explore the depth information provided in the dataset. Due to the differences between the HighFive dataset and the Hollywood3D dataset and lack of multiview and depth information, we do not expect to outperform state-of-art methods, but still we significantly outperform [8], doubling the performance in terms of average precision.

HighFive and Hollywood3D differ in many ways. Most importantly, the action categories differ significantly: while the trees are trained on HighFive to best discriminate *Hug* or *Kiss* from *High Five* and *Hand Shake*, on Hollywood3D they achieve promising performance in discriminating *Hug* or *Kiss* from 11 other actions such as *Eat* and *Use Phone* that the original tree discovering had no knowledge about. These results are further evidence for the generalization and discriminative power of the trees we discover, and show potential for re-use of the learned trees across datasets.

9. Conclusion

Our approach discovers a compact set of hierarchical space-time tree structures of human actions from training videos. Using an ensemble of the discovered trees, or in combination with simpler action words and pairwise structures, we build action classifiers that achieve state-of-the-art performance on two challenging datasets: HighFive and UCF-Sports. We also show cross-dataset generalization of the trees learned on HighFive on the Hollywood3D dataset.

Acknowledgments. This work was supported in part through US NSF grants #1029430 and #0910908.

References

[1] N. B. Aoun, M. Mejdoub, and C. B. Amar. Graph-based approach for human action recognition using spatio-temporal features. *Journal of Visual Communication and Image Representation*, 25(2):329 – 338, 2014. 2

[2] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011. 2

[3] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2001. 6

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008. 6

[5] P. F. Felzenszwalb and R. Zabih. Dynamic programming and graph algorithms in computer vision. *TPAMI*, 33(4):721–740, 2011. 3

[6] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, (315):972–976, 2007. 4, 6

[7] A. Gaidon, Z. Harchaoui, and C. Schmid. Activity representation with motion hierarchies. *IJCV*, 107(3):219–238, 2014. 2, 7

[8] S. Hadfield and R. Bowden. Hollywood 3D: Recognizing actions in 3D natural scenes. In *CVPR*, 2013. 8

[9] S. Hadfield, K. Lebeda, and R. Bowden. Natural action recognition using invariant 3D motion encoding. In *ECCV*, 2014. 8

[10] M. Hoai and A. Zisserman. Discriminative subcategorization. In *CVPR*, 2013. 6

[11] N. Ikizler and D. A. Forsyth. Searching for complex human activities with no visual examples. *IJCV*, 80(3):337–357, 2008. 2

[12] V. Kantorov and I. Laptev. Efficient feature extraction, encoding, and classification for action recognition. In *CVPR*, 2014. 2

[13] T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *ICCV*, 2011. 6, 7

[14] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 2, 7

[15] S. Ma, J. Zhang, N. Ikizler-Cinbis, and S. Sclaroff. Action recognition and localization by hierarchical space-time segments. In *ICCV*, 2013. 2, 4, 6, 7, 8

[16] P. Matikainen, M. Hebert, and R. Sukthankar. Representing pairwise spatial and temporal relations for action recognition. In *ECCV*, 2010. 2, 5

[17] S. Nijssen and J. N. Kok. A quickstart in frequent structure mining can make a difference. In *ICCS*, 2005. 4

[18] D. Oneata, J. Verbeek, and C. Schmid. Action and Event Recognition with Fisher Vectors on a Compact Feature Set. In *ICCV*, 2013. 2

[19] A. Patron-Perez, M. Marszalek, I. Reid, and A. Zisserman. Structured learning of human interactions in TV shows. *TPAMI.*, 34(12):2441–2453, 2012. 7

[20] A. Patron-Perez, M. Marszalek, A. Zisserman, and I. D. Reid. High five: Recognising human interactions in TV shows. In *BMVC*, 2010. 6

[21] D. Ramanan and D. A. Forsyth. Automatic annotation of everyday movements. In *NIPS*, 2003. 2

[22] M. Raptis, I. Kokkinos, and S. Soatto. Discovering discriminative action parts from mid-level video representations. In *CVPR*, 2012. 2, 4, 7

[23] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *CVPR*, 2013. 2

[24] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008. 6

[25] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. 2

[26] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *CVPR*, 2013. 7

[27] S. Todorovic. Human activities as stochastic kronecker graphs. In *ECCV*, 2012. 2

[28] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, pages 1–20, 2013. 2, 7

[29] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 2

[30] L. Wang, Y. Qiao, and X. Tang. Video action detection with relational dynamic-poselets. In *ECCV*, 2014. 2

[31] L. Wang and H. Sahbi. Directed acyclic graph kernels for action recognition. In *ICCV*, 2013. 2, 7

[32] Y. Wang and G. Mori. Learning a discriminative hidden part model for human action recognition. In *NIPS*, 2008. 2

[33] Y. Wang and G. Mori. Hidden part models for human action recognition: Probabilistic versus max margin. *TPAMI*, 33(7):1310–1323, 2011. 2, 3

[34] D. Weinland, E. Boyer, and R. Ronfard. Action recognition from arbitrary views using 3D exemplars. In *ICCV*, 2007. 2

[35] B. Wu, C. Yuan, and W. Hu. Human action recognition based on context-dependent graph kernels. In *CVPR*, 2014. 2

[36] X. Yang and Y. Tian. Action recognition using super sparse coding vector with spatio-temporal awareness. In *ECCV*, 2014. 2

[37] H. Zhang, W. Zhou, C. M. Reardon, and L. E. Parker. Simplex-based 3D spatio-temporal feature description for action recognition. In *CVPR*, 2014. 2