Real-time Online Action Detection Forests using Spatio-temporal Contexts

Seungryul Baek
Imperial College London
s.baek15@imperial.ac.uk

Kwang In Kim
University of Bath
k.kim@bath.ac.uk

Tae-Kyun Kim
Imperial College London
tk.kim@imperial.ac.uk

Abstract

Online action detection (OAD) is challenging since 1) robust yet computationally expensive features cannot be straightforwardly used due to the real-time processing requirements and 2) the localization and classification of actions have to be performed even before they are fully observed. We propose a new random forest (RF)-based online action detection framework that addresses these challenges. Our algorithm uses computationally efficient skeletal joint features. High accuracy is achieved by using robust convolutional neural network (CNN)-based features which are extracted from the raw RGBD images, plus the temporal relationships between the current frame of interest, and the past and futures *frames. While these high-quality features are not available in real-time testing scenario, we demonstrate that they can be effectively exploited in training RF classifiers: We use these* spatio-temporal contexts *to craft RF's new split functions improving RFs' leaf node statistics. Experiments with challenging MSRAction3D, G3D, and OAD datasets demonstrate that our algorithm significantly improves the accuracy over the state-of-the-art online action detection algorithms while achieving the real-time efficiency of existing skeleton-based RF classifiers.*

1. Introduction

Online action detection (OAD) refers to the task of simultaneously localizing actions and identifying their classes from *streamed* input sequences [11, 16, 26, 25, 52]. OAD facilitates a wide range of real-time applications including visual surveillance, pervasive health-care, and human-computer interaction. These new opportunities, however come with significant challenges not encountered by the classical *offline* action recognition approaches [49, 42, 12, 40, 47, 44, 43, 39, 7, 15, 19, 50, 5, 10]: Realistic OAD applications require real-time processing that prevents the use of computationally expensive features. Significant progress in this direction has been made with the recent emergence of consumer-level depth sensors: For instance, the Microsoft Kinect sensor enables real-time estimation of skeleton joints which provides highly descriptive and compact (*i.e*., 10 to 20 key-point coordinates of a human body) representations of complex human motions, and thereby facilitates subsequent real-time recognition. However, due to occlusions [31] and variations of appearance (of clothing, body shape, scale [37] *etc*.), estimated skeleton joints are often noisy [30, 32, 2, 51], leading to unreliable action recognition.

The state-of-the-art convolutional neural network (CNN)-based features [22, 6] extracted from RGB and depth images have demonstrated greater robustness in action recognition [34, 33, 8, 39]. In particular, they have been demonstrated to be effective in capturing the fine-grained attributes of human actions [18] (as well as other scene objects [17]) providing robustness to appearance variations [37, 52] often encountered in human action recognition. However, incorporating sophisticated CNN features in OAD frameworks is challenging due to the real-time processing requirement.

Another main challenge of OAD is that the intended action has to be localized and classified immediately after or even before the action is completed. This precludes exploiting temporal context of the current action, spanning the previous and the unobserved future frames [40, 14].

In this paper, we present a new framework that effectively incorporates the expensive CNN-based features as well as the temporal context present in the past and future frames. The key idea is to exploit these additional *spatio-temporal contexts* only in the *training* phase of the action detection system. Once trained, our system uses the computationally efficient skeleton features enabling online, real-time testing. We instantiate this idea in a novel random forest (RF) algorithm where the refined RF parameters (*i.e*. split functions and leaf node statistics) are learned with the aid of contexts. The use of RFs is motivated by the facts that 1) Each tree of RFs is computationally extremely efficient and independent (easily parallelizable), which make RFs suitable for real-time applications; 2) RF is highly flexible, accommodating multiple feature modalities (*e.g*. skeletons, CNN-based features, and time indices) and the heterogeneous training criteria (regression and classification) allow-

978-1-5090-4822-9/17 $31.00 © 2017 IEEE
DOI 10.1109/WACV.2017.25

ing the incorporation of both action localization (regression) and classification into a single unified RF framework.

Experimental comparison with eight state-of-the-art algorithms on three challenging datasets demonstrates that our new algorithm achieves the state-of-the-art performance while retaining the low computational complexity suitable for OAD.

Our contributions to the literature are summarized as:

- We propose to use RGBD-based CNN features as spatial contexts and temporal location features as temporal contexts to compensate for the noisy skeleton estimation and lack of sufficient spatio-temporal information in the OAD framework.
- We instantiate spatio-temporal contexts in RFs using the objective similar to the conditional random field. Instead of using the original objective, we divide them into independent multiple objectives and randomly mix them up sequentially in the RFs.
- We obtain real-time action detection forests by mapping all contextual information by improving split functions of RFs and resultant leaf node statistics at the training stage, which makes the contexts not used at the testing stage for the real-time efficiency.

2. Related work

This section reviews related work, categorized into two: 1) skeleton-based OAD; 2) random forests that exploit auxiliary information. Our RF framework can be seen as a particular case of the second category.

Skeleton-based online action detection. Skeleton-based frameworks [1, 52, 29, 35, 26, 25] have been used for the OAD problem, due to its efficiency and good performance. In [29], Nowozin *et al*. defined short sequences (*i.e*. 35-frame interval sequences) as *action points* based on skeleton joints and they showed that this basic unit and RF-based framework are promising for the online method. In [1], *action points* were annotated for the G3D dataset proposed in [2]. A similar notion of action points was utilized in [35] but with different-scale time intervals. The moving pose descriptor [52] achieved a good performance by capturing both the skeleton joints at one frame and the speed/acceleration of joints around the frame in a short sequence. In [26], the combination of the moving pose descriptor using the angles and the codebook model showed the promising results in real-time online action recognition. In [25], Li *et al*. introduced more realistic OAD datasets and baseline methods. They have proposed recurrent neural network (RNN)-based joint classification and regression framework using raw skeleton inputs and have shown the state-of-the-art results with relatively low time complexity. They reported that skeleton input is much attractive than RGBD [11] for OAD framework due to computational efficiency.

OAD is a challenging problem [11, 25] and even though skeleton-based approaches [1, 52, 29, 35, 26, 25] are promising, they are bounded by the time constraint that prevents combining multiple feature modalities or considering more temporal relationships. Dissimilar to the previous approaches [29, 35, 26, 25] based on a limited feature space, we try to explore multiple spatio-temporal feature spaces but only at the training stage, to improve the classification accuracy while keeping the real-time efficiency of the testing stage.

Training forests with auxiliary information. There have been several works [41, 53, 48, 21, 27, 38] that use the auxiliary information when training RFs other than the mapping between feature space to the label space. The auxiliary information is often captured by the interactions among data samples: some use pairwise interactions between data samples [41, 53, 38, 27] while others [48] use high-order interactions shared among more than two data samples. Also, there have been a method combining conditional random fields and RFs to use both pairwise and high-order interactions in a forest [21]. In [41], pairwise interactions were defined as links between corresponding synthetic and natural data samples to penalize separating them for the purpose of transfer learning. In [53], to deal with sparse and noisy information for the clustering problem, pairwise links were manually labeled between samples as hard constraints (*i.e*. must-link, cannot-link) and forests stopped growing when split functions violate the constraints. In [48], Yang *et al*. dealt with a face recognition problem by considering the consistency of head poses as a high-order interaction at the training stage. To treat continuous head poses in RFs, they discretized the head pose space into several discrete labels and formulated another classification task. In [38, 27], to incorporate both label co-occurrence and spatial consistency that are important high-order interactions for the segmentation task, new feature encoding method was designed for RF inputs. [27] is differentiated from [38] by performing it in an unified forest. In [21], as a generalized approach for [27, 38], new feature encoding method and a field-inspired split objective for RFs were proposed to incorporate the spatial consistency. However, Kontschieder *et al*. [21] reported that they did not fully use either pairwise or high-order interactions among samples when training RFs since it does not show the consistent improvement in classification accuracy. As a result, in [21], spatial consistency is come from feature encoding methods rather than the RF algorithms.

We train our RFs by the conditional random field-like objective to predict RFs' split parameters considering auxiliary information. Compared to previous algorithms, we

use multi-modal feature spaces to constitute the auxiliary information and fully model it when training RFs, with both pairwise and high-order interactions. Also, to secure the real-time testing time for our application, we use the auxiliary information at the training stage only.

3. Contexts in skeleton-based online action detection

Skeleton joints are commonly used as features in online action detection as they can be estimated real-time using widely available depth sensors [29, 35, 26, 52, 25]. We use a combination of skeletal joint positions and their time derivatives as proposed by Zanfir *et al*. [52]: A $9 \times n$-dimensional feature vector $\mathbf{x}(I) \in \mathcal{X}$ is formed at the input frame I by combining n joint positions (each consists of image (x and y) and depth (z) coordinate values) $\mathbf{p}(I)$ with their first- and second-order time derivatives: $\mathbf{x} = [\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}]^\top$.

Unfortunately, using only skeleton features can lead to erroneous classification as the estimated joint positions are often noisy. In this section, we introduce the spatio-temporal contexts that can complement the noisy skeleton features. Section 4 presents a new random forests model that effectively exploits these spatio-temporal contexts.

3.1. Spatio-temporal contexts

One of the biggest challenges in OAD, compared to traditional offline action recognition, is the lack of high-quality features on which classification can be based. For instance, in OAD information on future events is not available at the testing stage. Furthermore, due to the real-time processing requirement, computationally expensive features cannot be used. We propose to exploit these high-quality features to guide the *training* process.

Our spatio-temporal contexts are motivated by their effectiveness in offline action recognition [39, 8, 19, 10] (while other contexts are also possible). First, the *spatial context* is defined as the expensive yet powerful convolutional neural network (CNN)-based feature descriptor extracted from the holistic scene where human actions take place: A $8,192$-dimensional spatial context vector $\mathbf{z}^S \in \mathcal{Z}^S$ is constructed by combining two $4,096$-dimensional CNN feature vectors extracted from depth and RGB images, respectively. For the RGB map, we use Chatfield *et al*.'s *VGG-s* architecture [6] trained on the *ImageNet* dataset, while for the depth map, Rahmani and Mian's network trained on synthetic multi-view depth maps [34] is used.

Our *temporal context* is defined as the relative temporal location of a frame of interest in the entire action sequence. This feature is motivated by its ability to disambiguate similar-looking features: Some feature instances are inherently ambiguous as multiple action classes can exhibit them in the respective action sequences. However, these (ambiguous) feature instances are often observed at different time instances within the entire action sequences and therefore, putting them in this context can help reveal the underling action classes. Since this temporal context is not available at testing, it cannot be directly used in disambiguating similar features. Instead, we use it in training to put emphasis on refining decisions on the ambiguous feature instances in $\mathcal{X}$. The temporal context $\mathbf{z}^T(I_t) \in \mathcal{Z}^T$ of the t-th frame I_t in the given action sequence $\mathcal{A}$ is defined as follows:

$$\mathbf{z}^T(I_t) = \frac{t}{|\mathcal{A}|} \tag{1}$$

with $|\mathcal{A}|$ being the length (in the number of frames) of the action sequence $\mathcal{A}$.

4. Context-guided action detection forests

Our random forests (RFs) adopt two types of features: skeleton features $\mathbf{x}$ and spatio-temporal contexts $\mathbf{z} = \{\mathbf{z}^S, \mathbf{z}^T\}$. For testing, only skeleton features are used as the spatio-temporal contexts are not available, while during training (*i.e*. constructing the split functions and leaf node statistics of the RFs), RFs are provided with $\mathbf{z}$ enabling the exploitation of the rich contextual information therein.

Random forests. Random Forests (RFs) are an ensemble of binary decision trees that are constructed based on bootstrapped data. Each tree consists of two types of nodes: *split nodes* and *leaf nodes*. For an input data point $\mathbf{h}$, a split node conducts a simple binary test, *e.g*. by comparing the γ-th feature value $\mathbf{h}_\gamma$ of $\mathbf{h}$ with a certain threshold T.[1] Each out-branch of the split node then represents the outcome of the test, and $\mathbf{h}$ is accordingly sent to either the left or right child (*e.g*. left if $\mathbf{h}_\gamma \leq T$, right otherwise).

Training RFs involves growing the trees and partitioning the training set $\mathcal{D}$ accordingly, by deciding each node's decision behavior prescribed by the parameters (γ and T) of a *split function* $\Psi_{T,\gamma}(\mathbf{h}) := \text{sgn}(\mathbf{h}_\gamma - T)$. Once the input training (sub-)set $\mathcal{D}$ is arrived at a split node, a set of candidate split functions $\{\Psi^c\}$ is randomly generated and among them, the one that maximizes a split objective function $\mathcal{O}$ is selected as the final split function Ψ^*:

$$\Psi^* = \underset{\Psi \in \{\Psi^c\}}{\arg\min}\, \mathcal{O}(\Psi) \tag{2}$$

which divides $\mathcal{D}$ into disjoint subsets $\mathcal{D}_L$ and $\mathcal{D}_R$. The details of the objective $\mathcal{O}$ will be explained shortly.

Once the tree is generated, each leaf node stores the empirical distribution of the set of training data points arrived at that node. Therefore, at testing, the leaf node arrived

[1] For computational efficiency, we consider only simple single-feature comparisons while more complex split functions are also possible.

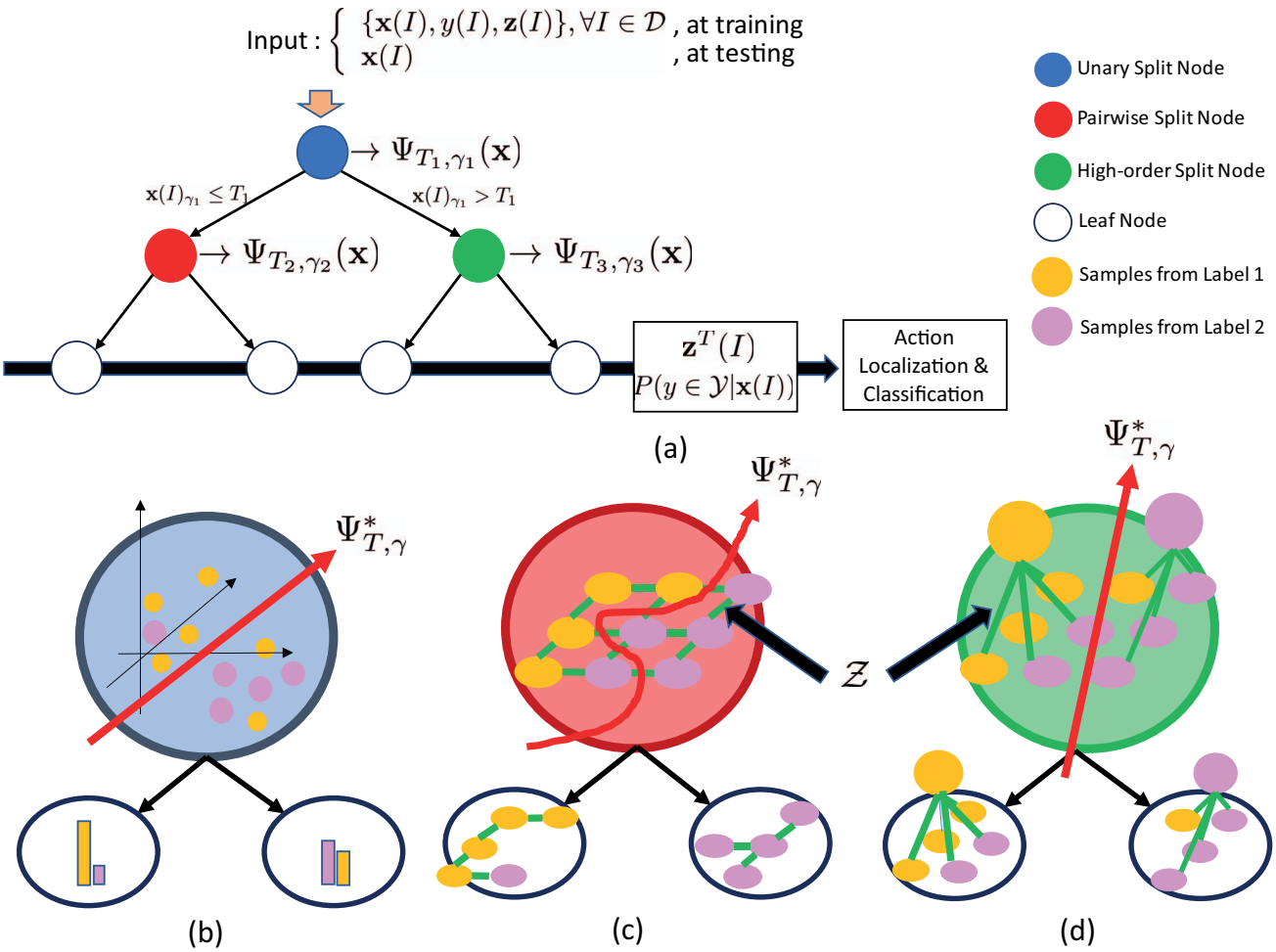


Figure 1. Schematic diagrams of the proposed decision tree architecture (Best viewed with color). Trained forests are given as an ensemble of individual decision trees. (a) An example tree: Each split function $\Psi(\mathbf{x})$ partitions the (skeleton) feature space $\mathcal{X}$ aided by the context features $\{\mathbf{z}\} \subset \mathcal{Z}$. The learned split functions depend only on $\mathbf{x} \in \mathcal{X}$ as the context feature $\mathbf{z}$ is not available in testing. The leaf nodes (white circles) store the empirical (class label) distributions $P(y \in \mathcal{Y}|\mathbf{x}(I))$ and the temporal location feature values $\mathbf{z}^T$ of the corresponding training set partitions (see Sec 4.1 for their usage). The training objective of a split function is composed of unary, pairwise, and high-order potentials as illustrated in (b), (c) and (d), respectively. Green lines in (c) and (d) represent the relationships between data entries in the context space $\mathcal{Z}$. (b) The unary objective O_u is the standard entropy-based objective, representing the training error; (c) The pairwise objective O_p evaluates the smoothness of the context space $\mathcal{Z}$ partition measured in the graph cut criteria; (d) The high-order split objective O_h takes into account the contextual relationships among more than two data points, by evaluating the sample deviations from the mean of each partition.

by traversing $\mathbf{h}$'s path from the root represents an estimate of the class-conditional distribution (or more generally, the posterior distribution *e.g.* for regression). The final classification result is obtained by taking a majority vote from the modes of individual tree distributions. The regression outputs are constructed in a similar manner.

Online action detection with random forests. A major advantage of using the spatio-temporal contexts over skeleton features comes from their higher discriminative capacity. The underlying rationale of using these contexts only in training RFs (as they are not available in testing) is that they can steer the construction of split functions to put more emphasis on making fine-grained decisions on *ambiguous* patterns. Our preliminary experiments have revealed that spatio-temporal contexts can be best exploited when they are directly used to refine the geometry of the underlying data space: Even if $\mathbf{x}(I_j)$ and $\mathbf{x}(I_k)$ are similar, they should belong to different tree branches when $\mathbf{z}(I_j)$ and $\mathbf{z}(I_k)$ are different. This observation is incorporated

into the new split objective $\mathcal{O}$ of our RFs, which is similar to the energy potential used in conditional random fields (CRFs) [13, 20, 23]:

$$\mathcal{O}(\Psi;\mathcal{Z}) = \underbrace{\mathcal{O}_u(\Psi)}_{unary} + \lambda_1 \underbrace{\mathcal{O}_p(\Psi;\mathcal{Z})}_{pairwise} + \lambda_2 \underbrace{\mathcal{O}_h(\Psi;\mathcal{Z})}_{high-order}, \quad (3)$$

where λ_1 and λ_2 are hyper-parameters that weight the contributions of different potentials:

$$\mathcal{O}_u(\Psi) = -\sum_{v\in\{L,R\}}\sum_{y\in\mathcal{Y}} n(y,\mathcal{D}_v)\log\frac{n(y,\mathcal{D}_v)}{|\mathcal{D}_v|}, \quad (4)$$

$$\mathcal{O}_p(\Psi;\mathcal{Z}) = \sum_{I_j\in\mathcal{D}_L}\sum_{I_k\in\mathcal{D}_R} w(I_j,I_k;\mathcal{Z}), \quad (5)$$

$$\mathcal{O}_h(\Psi;\mathcal{Z}) = \sum_{v\in\{L,R\}}\sum_{I_j\in\mathcal{D}_v}\left\|\mathbf{z}(I_j) - \frac{1}{|\mathcal{D}_v|}\sum_{I_l\in\mathcal{D}_v}\mathbf{z}(I_l)\right\|^2, \quad (6)$$

$n(y,\mathcal{D}_v)$ denotes the number of data points with class label y in $\mathcal{D}_v$, $\mathcal{D}_L$ and $\mathcal{D}_R$ are the left and right data splits, respectively as decided by the split function Ψ, and $w(I_j,I_k;\mathcal{Z})$ is the similarity between $\mathbf{z}(I_j)$ and $\mathbf{z}(I_k)$ as defined shortly (Eq. 7). Minimizing $\mathcal{O}$ (Eq. 3) for each split function Ψ enables the resulting decision tree to take account high-order information similarly to CRFs: It has been demonstrated that CRF's high-order potentials can help to identify the underlying structure in the output space [13, 20, 23] and thereby offer higher classification accuracy than pairwise CRFs in the presence of global interactions [3].

Typically in CRFs, a single unified potential $\mathcal{O}$ (Eq. 3) is directly optimized [13, 20, 23]. However, this requires explicitly determining the hyper-parameters λ_1 and λ_2. Recently, it has been demonstrated that for RFs, multiple heterogeneous split objectives can be straightforwardly incorporated, even without having to introduce the weighting hyper-parameters, by randomly sampling a sub-split objective at each node's splitting process [9, 41, 48]. We adopt this sampling approach to grow the decision trees: At each split point, a single split objective is randomly sampled from the set of five objective types $\mathcal{O}^{new}(\Psi;\mathcal{Z}) = \{\mathcal{O}_u(\Psi), \mathcal{O}_p(\Psi;\mathcal{Z}^S), \mathcal{O}_p(\Psi,\mathcal{Z}^T), \mathcal{O}_h(\Psi,\mathcal{Z}^S), \mathcal{O}_h(\Psi,\mathcal{Z}^T)\}$. Once the split objective is determined, a new split function Ψ^* is obtained as its minimizer from the candidate split functions $\{\Psi^c\}$. Note that pairwise and high-order potentials are defined for each context type. The overall training process is summarized in Algorithm 1. The remainder of this section presents the details of each split objective (Fig. 1).

Unary objective: mapping skeletons to action class. The unary objective $\mathcal{O}_u$ (Eq. 4) is same as the objective of the standard RFs classifiers: It evaluates the training error by measuring the uncertainty (or empirical entropy estimates) of class label distributions in the two child datasets $\mathcal{D}_L$ and $\mathcal{D}_R$ (Fig. 1b).

Algorithm 1: Training each split node with three split objectives $\mathcal{O} = \{\mathcal{O}_u, \mathcal{O}_p, \mathcal{O}_h\}$.

Input: $\mathcal{Z} = \{\mathcal{Z}^S, \mathcal{Z}^T\}$ and $\mathcal{X}$ for $\mathcal{D}$ arrived at each split node.
Output: Split function Ψ^*, Data splits $\mathcal{D}_L$ and $\mathcal{D}_R$ at each split node.
Generate a reference split Ψ^0 whose left child has all samples $\mathcal{D}_L = \mathcal{D}$ and right child has no sample $\mathcal{D}_R = \phi$.
Randomly generate split function candidates $\{\Psi^c\}$ and corresponding data splits $\mathcal{D}_L$ and $\mathcal{D}_R$.
Randomly select one of split objectives from $\mathcal{O}^{new}(\Psi;\mathcal{Z})$ defined in Sec 4.
Calculate the score $\mathcal{O}(\Psi)$ for $\{\Psi^c\}$ and Ψ^0 according to the selected split objective (Eqs. 4-6).
Find the Best Split function Ψ^* by Eq. 2.
Calculate *Information gain* $IG = \mathcal{O}(\Psi^*) - \mathcal{O}(\Psi^0)$.
if $IG \leq 0$ *on* Ψ^* **then**
  Make Leaf (*i.e.* Stop Growing).
end
else
  Split $\mathcal{D}$ into $\mathcal{D}_L$ and $\mathcal{D}_R$ using Ψ^*.
end

Pairwise objective: pair-wise context consistency. The pairwise objective $\mathcal{O}_p$ (Eq. 5) measures the smoothness of data points in each child dataset (Fig. 1c). This is essentially a graph cut criterion on $\mathcal{Z}$ and the splits made at each node corresponds to graph partitioning [36, 46]: At each split node, a parent dataset $\mathcal{D}$ is represented as a disjoint union of two child sets $\mathcal{D}_L$ and $\mathcal{D}_R$. Using the pairwise similarity w enforces that similar images I_j and I_k are grouped together when the corresponding contexts are similar:

$$w(I_j,I_k;\mathcal{Z}) = \exp\left(\frac{-\|\mathbf{z}(I_j) - \mathbf{z}(I_k)\|_2}{\sigma^2}\right) \quad (7)$$

with σ^2 being a parameter set as the mean of $\|\mathbf{z}(I_j) - \mathbf{z}(I_k)\|_2$ for $\forall I_j, I_k \in \mathcal{D}$. Directly optimizing $\mathcal{O}_p$ corresponds to solving the *minimum cut* [46] problem which has a bias towards small sets of isolated nodes [36]. Instead, we balance the grouping volumes by adopting the normalized cut (NCut) criteria [36]. For a normalized graph Laplacian L of a data set $\mathcal{D}$:

$$L = I - D^{-1/2}WD^{-1/2}, \quad (8)$$

with $W_{jk} = w(I_j,I_k;\mathcal{Z})$ and D being a diagonal matrix defined as $D_{ii} = \sum_j A_{ij}$, the eigenvector $\mathbf{e}$ corresponding

to the second smallest eigenvalue of L gives the solution of a continuous relaxation of NCut problem on $\mathcal{D}$ [36]. Partitioning the training set $\mathcal{D}$ can be achieved by thresholding each entry of $\mathbf{e}$. However, to be able to process a new test data instance, the partitioning criteria (or equivalently the split function) has to be explicitly represented as a function on $\mathcal{X}$. The resulting split function Ψ^* is obtained as

$$\Psi^* = \underset{\Psi \in \{\Psi^c\}}{\arg\min} \sum_{v \in \{L,R\}} \sum_{j \in \mathcal{D}_v} \|\mathbf{e}_j - \mu_v\|^2 \qquad (9)$$

where μ_v is the mean of $\{\mathbf{e}_j\}, \forall j \in \mathcal{D}_v$.

High-order objective: group context consistency. The high-order objective $\mathcal{O}_h$ (Eq.6) decides the split function Ψ^* that partitions data points into two *coherent* groups: The mean deviation of data points in each group is minimized. This coincides with the split criterion of regression RFs [41, 9, 28] (Fig. 1d).

4.1. Testing of OAD forests

Applying the trained OAD RFs to a new test image I is straightforward as the split functions $\{\Psi\}$ are expressed only with the skeleton features $\mathbf{x}$. We refine these initial classification results by taking the temporal consistency into account.

Initial action classification. For each input frame I, a feature vector $\mathbf{x}(I)$ is calculated and fed to the trained RFs. As shown in Fig. 1a, the corresponding leaf node of each tree stores the empirical class distribution $P(y \in \mathcal{Y}|\mathbf{x}(I))$ and the mean temporal context feature value $\mathbf{z}^T$. By averaging these empirical distributions and the mean temporal context values over the forests, we predict the initial class label of I and its temporal location feature $\mathbf{z}^T(I)$.

Action localization and refinement. The estimated temporal context $\mathbf{z}^T(I)$ of the input frame I gives an indication on how likely I is at the start or end of an action sequence: If $\mathbf{z}^T(I)$ is less than a small threshold value β (start point) or larger than $1-\beta$ (end point) and there is an action change between previous and current frames by the initial action classification result, I is marked as a changing point of an action. When a start frame is observed, the initial classification results are aggregated until the end frame is detected. Once the end frame is reached, the most probable action class is assigned for the detected interval. The threshold β is decided to minimize the training classification error similarly to [26].

5. Experiments

We compare performance of our algorithm with eight state-of-the-art action detection algorithms [52, 26, 35, 1, 25] on three datasets, *MSR Action3D* [24], *G3D* [2], and *OAD* [25]. These datasets cover slightly different OAD application scenarios: 1) without 'no action' frames, 2) with action point annotations, and 3) with start/end frame annotations, respectively.

Method	mean AP	Runtime per frame
Moving Pose [52]	0.890±0.002	–
ELS [26]	0.902±0.007	9.1 ms
RF	0.820±0.015	1.1 ms
RF+T	0.885±0.012	
RF+ST	$\mathbf{0.920 \pm 0.008}$	

Table 1. Performance of different action detection algorithms on *MSR Action3D* dataset.

Method	avg. F-score	Runtime per frame
Bloom *et al*. [1]	0.919	–
Multi-scale [35]	0.937	2.0 ms
RF	0.887	1.1 ms
RF+T	0.913	
RF+ST	**0.948**	

Table 2. Performance of different action detection algorithms on *Fighting* sequence of *G3D* dataset based on F-score at $\Delta = 333ms$ based on leave-one-out validation (see [1] for details).

Tables 1-3 summarize the results: *RF*, *RF+T*, and *RF+ST* denote our RF models without contexts, with only temporal contexts, and with spatio-temporal contexts, respectively. Since our baseline *RF* model does not perform action localization, we adopt the the sliding window approach of [52, 25] and determine the window size based on cross-validation. For all three algorithms, the diversity of RFs is maximized by applying bootstrapping [4], *i.e*. each tree is trained with a different training subset. To address OAD's class-imbalance problem (due to the larger 'no-action' class sample size than the remaining 'action' classes), each training subset is sampled from a weighted distribution inversely proportional to the (sample) size of each class. In the remainder of this section, we present the details of the experimental settings and the evaluation protocol for each dataset.

5.1. MSR Action3D dataset: w/o 'no-action' frames

The *MSR Action3D* dataset includes 20 actions performed by 10 subjects and it provides depth maps and 20 skeleton joints captured by Kinect camera. This dataset was originally constructed for the offline action classification scenario [24]; thus each video is pre-segmented to contain only one action class. Zanfir *et al*. [52] combined these individual 557 video sequences to a single long video which enables to simulate an OAD problem. However this new dataset is limited in that no negative (or 'no action') frames are included. We adopt the same experimental settings and

Actions	*SVM-SW* [25]	*RNN-SW* [45]	*CA-RNN* [25]	*JCR-RNN* [25]	*RF*	*RF+T*	*RF+ST*
drinking	0.146	0.441	**0.584**	0.574	0.253	0.298	0.517
eating	0.465	0.550	0.558	0.523	0.661	**0.662**	0.645
writing	0.645	**0.859**	0.749	0.822	0.761	0.858	0.803
opening cupboard	0.308	0.321	0.490	0.495	0.427	0.478	**0.555**
washing hands	0.562	0.668	0.672	0.718	0.678	0.860	**0.860**
opening microwave	0.607	0.665	0.468	**0.703**	0.561	0.567	0.610
sweeping	0.461	0.590	0.597	**0.643**	0.224	0.273	0.437
gargling	0.437	0.550	0.579	0.623	0.383	0.368	**0.722**
throwing trash	0.554	0.674	0.430	0.459	0.626	0.671	**0.688**
wiping	0.857	0.747	0.761	0.780	0.916	0.948	**0.977**
Overall F-score	0.540	0.600	0.596	0.653	0.548	0.592	**0.672**
Runtime per sequence	**1.05**	3.14	–	2.60	1.60		
SL−score	0.316	0.366	0.378	0.418	0.320	0.356	**0.445**
EL−score	0.325	0.376	0.382	**0.443**	0.342	0.367	0.432

Table 3. Performance different action detection algorithms on *OAD* dataset.

evaluation protocols as in [26, 52] to facilitate a fair comparison with these algorithms and to gain an insight into the performance of our models on pure action recognition problems. Since the dataset does not provide RGB frames, our models use only depth sequences to constitute the spatial contexts $\{\mathbf{z}^S\} \subset \mathcal{Z}^S$. The experiments were performed on 100 random video combinations following [52], and the mean accuracies and standard variations were measured.

Table 1 summarizes the results. Our baseline model *RF* is not as competitive as the two state-of-the-art approaches [52, 26]. However, exploiting the temporal context (*RF+T*) already improved the accuracy by 7.3% and it is on par with these algorithms. Combining the spatial and temporal contexts further improved performances, surpassing [52, 26] with large margins.

5.2. G3D dataset : action point annotation

The *G3D* dataset [2] provides *action point* annotations. An action point is a single time instance sampled within the time span of an action [29, 35] facilitating an assessment of action detection performance. We adopt the evaluation protocols and the training and testing splits from [1, 35]. In this dataset, RGB and depth frames, and 20 skeleton joint positions are provided (as captured by the Kinect V1 camera). We observed that the provided depth maps are well aligned with the skeleton sequences while the RGB frames are not perfectly synchronized. Each time an RGB frame is missing, we use the latest available RGB frame to constitute the spatial context $\mathbf{z}^S$.

Table 2 summarizes performances of our models and the algorithms of [35, 1]. Similarly to *MSR Action3D* dataset case, our *RF* baseline showed the worst performance; however exploiting the spatio-temporal contexts significantly improved the performance of RFs resulting in the highest accuracy.

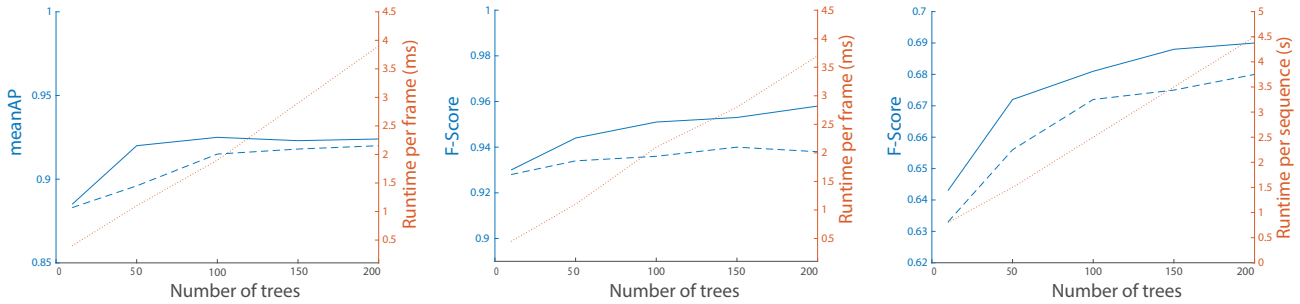
5.3. OAD dataset : start/end frame annotation

The *OAD* dataset includes 59 video clips of multiple action sequences plus *start / end frame* annotations [25]. The dataset provides RGB frames, depth sequences, and the tracked 25 skeleton joint positions captured by the Kinect V2 camera. We follow the evaluation protocols and the training and testing splits of Li *et al*. [25]: The classification accuracy is assessed based on the class-wise *F-score* as well as the overall F-score [25] while the accuracy of the start and end frame detection is measured in *SL-* and *EL-scores* as in [25].

Table 3 shows the performance of our models, Li *et al*.'s two recurrent neural network (RNN)-based algorithms (*RNN-SW* and *JCR-RNN*) [25], the baseline sliding window-based classifier (*SVM-SW*) [25], and Wang *et al*.'s algorithm *RNN-SW* [45] as reported in [25]. The performance of *RF* is similar to *SVM-SW* as both algorithms use sliding windows without explicitly detecting action changes. Our *RF+T* model improved upon them and achieved a similar performance to *RNN-SW* model of [45] as both are designed to incorporate the temporal information. Finally, our *RF+ST* model achieved the best overall F- and SL-scores and it showed the second best EL-score deviating from the best EL-score of *JCR-RNN* model [25] by only 2.5%. It should be noted that the feature space of *JCR-RNN* is limited to skeletons for real-time efficiency [25]. Even though RFs are computationally much more efficient than RNNs (Table 3), incorporating the spatio-temporal contexts enabled to achieve even better overall performance than RNNs.

5.4. Time complexity

Tables 1–3 report the runtimes of our models and the state-of-the-art algorithms [25, 26, 35, 45] measured on



(a) MSRAction3D dataset (b) G3D dataset (c) OAD dataset

Figure 2. The effect of tree numbers on accuracy and time complexity of random forest models. The blue and red curves correspond to the accuracy and the time complexity with respect to the number of trees, respectively. The dotted and solid blue curves show the RFs' accuracy without and with high-order potentials, respectively.

a machine with 6-core 2.84Hz CPU and 32GB of RAM, demonstrating the high computational efficiency of our algorithm. It should be noted that the runtimes of our three RF models (*RF*, *RF+T*, and *RF+ST*) are identical in testing as they all use only the skeleton features. This demonstrates that exploiting the contextual information only at training stage keeps the run-time efficiency while significantly improving the accuracy of RFs. The efficiency of our approach can be attributed to 1) the proven low latency of the skeleton feature representation [52]; 2) the inherent simplicity of RFs: RFs' testing time complexity is $O(FD)$ ($F = 50$ and $D = 100$; see Sec. 5.5). In the worst case, our RFs execute FD *if-* statements for parsing an input feature. This is significantly more efficient than CNNs involving many convolution operations.

The runtimes of Meshry *et al*. [26] and Sharar *et al*. [35] (Tables 1 and 2) are measured using the code shared by the authors of each paper. The computational complexities of the algorithms of [25] and [45] (Table 3) are provided in [25] where the runtimes are measured in average second per video clips. The reported runtime of our algorithm in this table is normalized based on our implementation of *SVM-SW* [35]: We first measured the actual runtimes of our algorithm and our *SVM-SW* implementation on a 2.84Hz CPU machine. Then, the reported runtime of our algorithm in Table 3 was calculated by multiplying the ratio between the two measured runtimes by the runtime of *SVM-SW* provided in [25].

5.5. Hyper-parameters

For our RF models, the number of trees F is set to 50. Each tree is grown until either the maximum tree depth $D = 100$ is reached, or for each split node, the minimum sample number 1 is reached or the information gain is not positive. Fig. 2 shows the effect of the tree numbers on the accuracy (blue curves in Fig. 2) and runtime of RF models (red dotted curves in Fig. 2). Overall, both accuracy and computational complexity increase monotonically with F while the steepness of the accuracy curves peak at around 50 trees suggesting the point of balanced trade between accuracy and computational complexity. We also observed that using high-order potentials $\mathcal{O}_h$ (Eq. 6) consistently improves the action recognition performance (blue dotted curves in Fig. 2).

6. Conclusion

Skeleton-based online action detection framework is promising due to its efficiency and relatively good performance; however it is not straightforward to further consider multiple spatio-temporal feature spaces in the framework, due to the online processing constraint. We use the RFs based on skeleton joints as our baseline approach since the algorithms are extremely efficient and thus, they are suitable for the real-time OAD problem. Inspired by the success of offline action recognition methods, we define two promising spatial and temporal contexts based on the convolutional neural network feature from RGBDs and the temporal location feature, respectively. The efficiency of our method is maintained while incorporating these contexts, since contexts are employed only at the training stage and we do not use any contexts at testing stages by mapping all the contexts into the split functions of RFs (Split functions are defined based only on the skeleton joints). We propose to utilize the conditional random field-like objective to incorporate both spatial and temporal contexts in such a way. Without introducing additional weighting hyper-parameters, we divide the original split objective into three small portions and mix them randomly and sequentially in the RFs. By experimenting with three challenging online action detection benchmarks (*i.e*. *MSR Action3D*, *G3D* and *OAD* datasets), we show that our RF models, trained with spatio-temporal contexts significantly improve the accuracy with low time complexity.

References

[1] V. Bloom, V. Argyriou, and D. Makris. Dynamic feature selection for online action recognition. *Human Behavior Understanding*, 2013.

[2] V. Bloom, D. Makris, and V. Argyriou. G3D: A gaming action dataset and real time action recognition evaluation framework. In *CVPR Workshop*, 2012.

[3] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, 2001.

[4] L. Breiman. Bagging predictors. *Machine Learning*, 1996.

[5] H. Chang, G. Garcia-Hernando, D. Tang, and T.-K. Kim. Spatio-temporal hough forest for efficient detection-localisation-recognition of fingerwriting in egocentric camera. *CVIU*, 2016.

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: delving deep into convolutional nets. In *BMVC*, 2014.

[7] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015.

[8] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

[9] J. Gall, A. Yao, N. Razavi, L. V. Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *TPAMI*, 2011.

[10] G. Garcia-Hernando, H. Chang, I. Serrano, O. Deniz, and T.-K. Kim. Transition hough forest for trajectory-based action recognition. In *WACV*, 2016.

[11] R. D. Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars. Online action detection. In *ECCV*, 2016.

[12] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015.

[13] J. M. Gonfaus, X. Boix, J. V. D. Weijer, A. D. Bagdanov, J. Serrat, and J. Gonzalez. Harmony potentials for joint classification and segmentation. In *CVPR*, 2010.

[14] A. Graves, S. Fernandez, and J. Schmidhuber. Bidirectional LSTM networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and Their Applications (ICANN)*, 2005.

[15] J.-F. Hu, W.-S. Zheng, J. Lai, and J. Zhang. Jointly learning heterogeneous features for RGB-D activity recognition. In *CVPR*, 2015.

[16] J.-F. Hu, W.-S. Zheng, L. Ma, G. Wang, and J. Lai. Real-time RGB-D activity prediction by soft regression. In *ECCV*, 2016.

[17] J. Huang, R. S. Feris, Q. Chen, and S. Yan. Cross-domain image retrieval with a dual attribute-aware ranking network. In *ICCV*, 2015.

[18] F. S. Khan, R. M. Anwer, J. van de Weijer, M. Felsberg, and J. Laaksonen. Deep semantic pyramids for human attributes and action recognition. In *SCIA*, 2015.

[19] T.-K. Kim and R. Cipolla. Canonical correlation analysis of video volume tensors for action categorization and detection. *TPAMI*, 2009.

[20] P. Kohli, L. Ladicky, and P. H. Torr. Robust higher order potentials for enforcing label consistency. In *CVPR*, 2008.

[21] P. Kontschieder, P. Kohli, J. Shotton, and A. Criminisi. GeoF: geodesic forests for learning coupled predictors. In *CVPR*, 2013.

[22] A. Krizhevsky, Sutskever, Ilya, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[23] L. Ladicky, C. Russell, P. Kohli, and P. Torr. Associative hierarchical CRFs for object class image segmentation. In *ICCV*, 2009.

[24] W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3D points. In *CVPR Workshop*, 2010.

[25] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, and J. Liu. Online human action detection using joint classification-regression recurrent neural networks. In *ECCV*, 2016.

[26] M. Meshry, M. E. Hussein, and M. Torki. Linear-time online action detection from 3D skeletal data using bags of gesturelets. In *WACV*, 2016.

[27] A. Montillo, J. Shotton, J. Winn, J. Iglesias, D. Metaxas, and A. Criminisi. Entangled decision forests and their application for semantic segmentation of CT images. In *Information Processing in Medical Imaging (IPMI)*, 2011.

[28] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, 2006.

[29] S. Nowozin and J. Shotton. Action points: a representation for low-latency online human action recognition. *Microsoft Research Cambridge, Technical Report*, 2012.

[30] O. Oreifej and Z. Liu. HON4D: histogram of oriented 4D normals for activity recognition from depth sequences. In *CVPR*, 2013.

[31] U. Rafi, J. Gall, and B. Leibe. A semantic occlusion model for human pose estimation from a single depth image. In *CVPR*, 2015.

[32] H. Rahmani, A. Mahmood, D. Q. Huynh, and A. Mian. HOPC: histogram of oriented principal components of 3D pointclouds for action recognition. In *ECCV*, 2014.

[33] H. Rahmani and A. Mian. Learning a non-linear knowledge transfer model for cross-view action recognition. In *CVPR*, 2015.

[34] H. Rahmani and A. Mian. 3D action recognition from novel viewpoints. In *CVPR*, 2016.

[35] A. Sharaf, M. Torki, M. E. Hussein, and M. El-Saban. Real-time multi-scale action detection from 3D skeleton data. In *WACV*, 2015.

[36] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.

[37] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. In *CVPR*, 2011.

[38] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[39] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[40] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-Stream bi-directional recurrent neural network for fine-grained action detection. In *CVPR*, 2016.

[41] D. Tang, T.-H. Yu, and T.-K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *ICCV*, 2013.

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

[43] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3D skeletons as points in a lie group. In *CVPR*, 2014.

[44] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Learning actionlet ensemble for 3D human action recognition. *TPAMI*, 2014.

[45] Y. Weiss, A. Torralba, and R. Fergus. Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In *AAAI*, 2016.

[46] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *TPAMI*, 1993.

[47] L. Xia, C.-C. Chen, and J. K. Aggarwal. View invariant human action recognition using histograms of 3D joints. In *CVPR Workshop on Human Activity Understanding from 3D Data*, 2012.

[48] H. Yang and I. Patras. Privileged information-based conditional structured output regression forest for facial point detection. *TCSVT*, 2015.

[49] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.

[50] T.-H. Yu, T.-K. Kim, and R. Cipolla. Real-time action recognition by spatiotemporal semantic and structural forests. In *BMVC*, 2010.

[51] T.-H. Yu, T.-K. Kim, and R. Cipolla. Unconstrained monocular 3D human pose estimation by action detection and cross-modality regression forest. In *CVPR*, 2013.

[52] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The moving pose: an efficienct 3D kinematics descriptor for low-latency action recognition and detection. In *ICCV*, 2013.

[53] X. Zhu, C. C. Loy, and S. Gong. Constrained clustering: effective constraint propagation with imperfect oracles. In *ICDM*, 2013.