Fast Action Detection via Discriminative Random Forest Voting and Top-K Subvolume Search

Gang Yu, *Student Member, IEEE*, Norberto A. Goussies, Junsong Yuan, *Member, IEEE*, and Zicheng Liu, *Senior Member, IEEE*

***Abstract*—Multiclass action detection in complex scenes is a challenging problem because of cluttered backgrounds and the large intra-class variations in each type of actions. To achieve efficient and robust action detection, we characterize a video as a collection of spatio-temporal interest points, and locate actions via finding spatio-temporal video subvolumes of the highest mutual information score towards each action class. A random forest is constructed to efficiently generate discriminative votes from individual interest points, and a fast top-K subvolume search algorithm is developed to find all action instances in a single round of search. Without significantly degrading the performance, such a top-K search can be performed on down-sampled score volumes for more efficient localization. Experiments on a challenging MSR Action Dataset II validate the effectiveness of our proposed multiclass action detection method. The detection speed is several orders of magnitude faster than existing methods.**

***Index Terms*—Action detection, branch and bound, random forest, top-K search.**

I. Introduction

UNDERSTANDING human behaviors is one of the core problems in many video-based applications, such as video surveillance, event-based video indexing and search, and intelligent human–computer interaction. Despite extensive studies in human action recognition and categorization [3], [4], [6], the detection and accurate localization of human actions remains a challenging problem. Different from action categorization, which only requires identifying which type of action occurs in a video clip, action detection needs to identify not only the occurrences of a specific type of actions but also where (spatial location in each frame) and when (temporal location) it occurs in the video [7]–[10]. An example is the detection of a person waving hands in a crowded and dynamic scene. It is in general a much more useful and challenging problem than categorization.

To robustly detect human actions, some early methods rely on the tracking of human bodies. With an accurate tracking of a human body and its movements, one can recognize and detect actions. However, this category of methods is of limited use in real applications, because reliable body tracking remains a difficult problem in crowded and dynamic scenes. For example, in a supermarket with many pedestrians, it is very difficult to detect and track all of the people, let alone to recognize their actions, e.g. someone raising his/her hands.

Instead of tracking human bodies, some other methods treat videos as spatio-temporal 3-D data and solve action detection using spatio-temporal template matching (3-D matching). Similar to the sliding window-based object detection, given an action template, the re-concurrences of the query action can be found by evaluating all of the possible video subvolumes. Despite previous successes of this approach, there are still two major challenges.

First of all, in the template matching method, usually only a single template is provided to perform action detection [21], [41]. In such a case, a single template cannot well characterize the intra-class variations of an action and is not discriminative enough for classification. Second, different from object detection, the search space in the spatio-temporal video space is extremely large. It thus greatly increases the computational cost for these template based approaches. For example, it is very time consuming to search actions of different spatial scales and different temporal durations in the video space. Although the recently proposed spatio-temporal branch-and-bound search method [9], [12] can significantly improve the search speed, it is still not fast enough to handle high-resolution videos (e.g. 320×240 and higher). Considering that the spatial localization is computationally more demanding for high-resolution videos, it is important to provide efficient solutions for high-resolution videos. Moreover, given a video dataset containing multiple action instances, it is desirable to efficiently detect all of them in one round of search.

To address the above challenges in action detection, we propose a random forest-based template matching method to detect actions, as shown in Fig. 1. Without performing background subtraction and human body tracking, each video sequence is characterized by a collection of spatio-temporal interest points (STIPs). During the training phase, a random forest is built to model the distribution of the STIPs from both positive and negative classes in the high-dimensional feature space. During the testing phase, each individual point matches the query class through the pre-built random forest, and provides an individual voting score toward each action type. Following the mutual information maximization formulation in [9], action detection be-

Manuscript received November 01, 2010; revised February 25, 2011; accepted March 03, 2011. Date of publication March 14, 2011; date of current version May 18, 2011. The work of J. Yuan was supported in part by the Nanyang Assistant Professorship SUG M58040015. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Chia-Wen Lin.

G. Yu and J. Yuan are with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639798 Singapore (e-mail: gyu1@e.ntu.edu.sg; jsyuan@ntu.edu.sg).

N. A. Goussies is with the Universidad de Buenos Aires, C1428EGA Buenos Aires, Argentina (e-mail: ngoussie@dc.uba.ar).

Z. Liu is with Microsoft Research Redmond, WA 98052-6399 USA (e-mail: zliu@microsoft.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2011.2128301

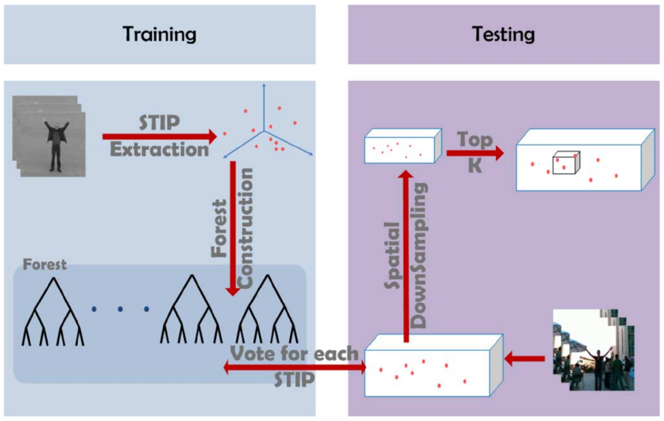


Fig. 1. Overview of our random forest-based video subvolume search.

comes finding the spatio-temporal video subvolume with the maximum total mutual information score.

Compared with the nearest-neighbor-based matching scheme in [9], our proposed random forest-based approach enables a much more efficient interest point matching without degrading the matching quality. Meanwhile, as both positive and negative action samples are taken into account while building the random forest, our proposed method not only handles intra-class action variations well, but also provides more discriminative matching to detect action instances. To reduce the computational overhead in searching high-resolution videos, we improve the original spatio-temporal branch-and-bound search method in [9] on two aspects. First of all, instead of performing branch-and-bound search in the original score volume, we propose to search a down-sampled score volume for efficient action localization. Our theoretical analysis shows that the error between the optimal solution of the down-sampled volume and that of the original volume can be upper bounded. Second, we propose a top-K search method to enable the detection of multiple action instances simultaneously in a single round of branch-and-bound search. It provides an efficient solution for multiclass multiple instance action detection.

To evaluate the efficiency and generalization ability of our proposed method, we perform a cross-dataset action detection test: our algorithm is trained on the KTH dataset and tested on the MSR action dataset II, which contains 54 challenging video sequences of both indoor and outdoor scenes. The extensive multiclass action detection results show that, ignoring the feature extraction cost, our proposed method can search a 1-h 320 $\times$ 240 video sequence in less than half an hour. It can detect actions of varying spatial scales and can well handle the intra-class action variations including performing style and speed variations and even partial occlusions. It also can handle cluttered and dynamic backgrounds. The proposed top-K volume search algorithm is general and can be used for any other applications of video pattern search.

II. Related Work

Even though there has been a large body of work in action categorization [30], [31], [33], [34], [39], [42], action detection is much less addressed in the literature. Different from action categorization, action detection is more challenging as it needs to locate the actions both spatially and temporally in cluttered or dynamic backgrounds [7]–[10], [12], [19], [20], [40]. Even with a good action categorization scheme, it can be time-consuming to search the video space and accurately locate the action. Compared with event detection [28], [29], [32], [43], action detection focuses on the various activities and movements performed by humans, which has a wide range of potential applications in our daily life.

There are mainly two types of existing approaches for action detection. The first is the template-based pattern matching [7], [11], [13], [21], [35]. Two types of temporal templates are proposed in [11] for characterizing actions: 1) the motion energy image (MEI), which is a binary image recording where the motion has occurred in an image sequence, and 2) the motion history image (MHI), which is scalar-valued image whose intensity is a function of the recent motion. In [14], the motion history volume (MHV) is introduced as a free-viewpoint representation for human actions. To better handle the cluttered and dynamic backgrounds, an input video is over-segmented into many spatio-temporal video volumes in [8]. An action template is matched by searching among these over-segmented video volumes. However, because only one template is utilized, previous template-based methods usually have difficulties in handling intra-class action variations. Some discriminative methods have been developed to improve template matching. In [15] and [16], Haar features are extended to 3-D space, and boosting is applied to integrate these features for final classification. In [18], a successive convex matching scheme is proposed for action detection. In [17], a prototype-based approach is introduced, where each action is treated as a sequence of prototypes. However, the computational costs of these algorithms are extremely high. For example, it takes several minutes or even hours to handle a single short video clip. More specifically, due to the need to enumerate all the possible subvolumes in a video clip, the computational complexity grows rapidly as the templates become more complex.

The second strategy is the tracking-based action detection. It relies on human tracking to provide a subvolume for categorization. In [10], a multiple-instance learning-based approach is applied to human action detection, which relies on head detection and tracking. Similarly, the techniques presented in [37] and [38] also require human tracking as a preprocessing step.

Generally speaking, both the template-based and tracking-based detection approaches have their own limitations. The tracking-based approach is largely constrained by the tracking precision. Since human tracking in a complex and dynamic environment is itself a challenging problem, it is not practical to rely on tracking to solve the action detection problem. On the other hand, the template-matching-based approach is usually computationally intensive. Although the branch-and-bound search proposed in [9] can speed up the action detection, the computational cost is still very high for high-resolution videos (such as 320 $\times$ 240 or higher) due to the large search space. Thus, a more efficient algorithm is required. Moreover, the template mainly relies on positive samples thus is not discriminative.

III. Multiclass Action Recognition

A. Mutual Information-Based Classification

We represent an action as a collection of STIPs [1], where $d \in \mathbb{R}^N$ denotes an N-dimensional feature vector describing a STIP. The reasons to represent the videos with STIP are its superior performance and that the HOG&HOF description is suitable for our random forest-based framework. A comparison of different detectors and descriptors can be seen in [49]. Denote the class label set as $\mathcal{C} = \{1, 2, \ldots, C\}$.

In order to recognize different action classes, we evaluate the pointwise mutual information[1] [45] between a testing video clip $\mathcal{Q} = \{d_q\}$ and one action class $c \in \mathcal{C}$ as

$$\begin{aligned} MI(\mathbf{C} = c, \mathcal{Q}) &= \log \frac{P(\mathcal{Q}|\mathbf{C} = c)}{P(\mathcal{Q})} \\ &= \log \frac{\prod_{d_q \in \mathcal{Q}} P(d_q|\mathbf{C} = c)}{\prod_{d_q \in \mathcal{Q}} P(d_q)} \\ &= \sum_{d_q \in \mathcal{Q}} \log \frac{P(d_q|\mathbf{C} = c)}{P(d_q)} \end{aligned} \tag{1}$$

where d_q refers to the STIP point in $\mathcal{Q}$ and we assume that d_q is independent of each other. Each $s^c(d_q) = log(P(d_q|\mathbf{C} = c)/P(d_q))$ is the pointwise mutual information between a STIP point d_q and a specific class c.

In the previous work [9], $s^c(d_q)$ is computed as follows:

$$s^c(d_q) = \mathrm{MI}(\mathbf{C} = c, d_q) = log \frac{C}{1 + \frac{P(d_q|\mathbf{C} \neq c)}{P(d_q|\mathbf{C} = c)}(C-1)} \tag{2}$$

where C is the number of classes. The likelihood ratio in (2) is calculated as

$$\frac{P(d_q|\mathbf{C} \neq c)}{P(d_q|\mathbf{C} = c)} \approx \lambda^c exp^{-\frac{1}{\sigma^2}\left(\|d_q - d_{\mathrm{NN}}^{c-}\|^2 - \|d_q - d_{\mathrm{NN}}^{c+}\|^2\right)} \tag{3}$$

where d_{NN}^{c+} and d_{NN}^{c-} are the nearest neighbors of d_q in the positive class and negative class, respectively, and λ^c is the ratio of the number of positive STIPs to the number of negative STIPs in the training dataset.

Despite its good performance, (3) has two limitations, which are given here.

- In order to calculate the likelihood ratio in (3), we need to search the nearest neighbors d_{NN}^{c+} and d_{NN}^{c-}. Although locality sensitive hash (LSH) has been employed for fast nearest-neighbor search, it is still time consuming for a large high-dimensional dataset.
- Only two STIPs are used to approximate the likelihood ratio in (3), which is not accurate.

To address the two problems, we reformulate the voting score $s^c(d_q)$ in (2) as

$$\begin{aligned} s^c(d_q) &= MI(\mathbf{C} = c, d_q) = \log \frac{P(d_q|\mathbf{C} = c)}{P(d_q)} \\ &= \log \frac{P(\mathbf{C} = c, d_q)}{P(\mathbf{C} = c)P(d_q)} \\ &= \log \frac{P(\mathbf{C} = c|d_q)}{P(\mathbf{C} = c)} \\ &= \log P(\mathbf{C} = c|d_q) - \log P(\mathbf{C} = c). \end{aligned} \tag{4}$$

As $P(\mathbf{C} = c)$ is a constant prior, the problem boils down to computing the posterior $P(\mathbf{C} = c|d_q)$. To enable an efficient computation, we approximate this probability with a random forest.

[1][Online]. Available: http://en.wikipedia.org/wiki/Pointwise_mutual_information

B. Random Forest-Based Voting

Random forest was first proposed to solve the classification problem [24]. Later, it was extended to handle regression problems and is used for many multimedia applications, as in [5], [6], [25]–[27], [36], and [46]. In our paper, random forest is employed to estimate the posterior probability $P(\mathbf{C} = c|d_q)$.

To build the forest from a training dataset, we use a method motivated by [5]. However, compared with [5], which treats a random forest as a classifier and votes for the hypothesis given a feature point, our random forest is used to estimate the posterior distribution of each STIP point.

Two kinds of descriptors for STIP: histogram of gradient (HOG) and histogram of flow (HOF), are used to build the random forest. In the following, we first describe how to build a single decision tree, and then the forest is constructed by M independent trees. Assume we have N STIP points in the training set, defined as $\{(x_i, y_i), i = 1, 2, \cdots, N\}$, where $x_i = (x_i^1, x_i^2)$; $x_i^1 \in R^{72}$ and $x_i^2 \in R^{90}$ refer to the HoG feature and HoF feature, respectively; $y_i \in \mathcal{C}$ is the label of the STIP (if we want to detect actions from category $\mathbf{C} = c$, we consider STIPs with $y_i = c$ as positive examples and other STIPs as negative examples). In order to build a tree and split the training set, a random number $\tau \in \{1, 2\}$ is first generated to indicate which kind of feature to use for splitting ($x_i^{\tau=1}$ refers to the HOG feature and $x_i^{\tau=2}$ refers to the HOF feature). Then, two more random integer numbers e_1 and e_2 will be generated, indicating the dimension indices of either HOG or HOF feature. After that, a "feature difference" can be evaluated with $D_i = x_i^\tau(e_1) - x_i^\tau(e_2), i = 1, 2, \cdots, N$. For each x_i, we assign it to the left child node if $x_i^\tau(e_1) - x_i^\tau(e_2) \geq \theta$ or right child node if $x_i^\tau(e_1) - x_i^\tau(e_2) < \theta$.

The threshold θ is selected by minimizing the binary classification error

$$\theta^* = \mathrm{argmin}_\theta \left(min\left\{\mathcal{E}(c)^L + \mathcal{E}(\bar{c})^R, \mathcal{E}(c)^R + \mathcal{E}(\bar{c})^L\right\}\right) \tag{5}$$

where

$$\begin{aligned} \mathcal{E}(c)^L &= \sum_{i=1}^N I(y_i \neq c) I\left(x_i^\tau(e_1) - x_i^\tau(e_2) \geq \theta\right) \\ \mathcal{E}(c)^R &= \sum_{i=1}^N I(y_i \neq c) I\left(x_i^\tau(e_1) - x_i^\tau(e_2) < \theta\right) \\ \mathcal{E}(\bar{c})^L &= \sum_{i=1}^N I(y_i = c) I\left(x_i^\tau(e_1) - x_i^\tau(e_2) \geq \theta\right) \\ \mathcal{E}(\bar{c})^R &= \sum_{i=1}^N I(y_i = c) I\left(x_i^\tau(e_1) - x_i^\tau(e_2) < \theta\right). \end{aligned} \tag{6}$$

In (6), $I(x)$ is a indicator function, that is, $I(x) = 1$ if $x = 1$ and 0 otherwise. Also, c is the action type we want to detect. The first two terms refer to the misclassification errors of the left and right nodes, respectively, when the labels of the nodes are both c. The last two terms refer to the misclassification errors of the left and right nodes, respectively, when the labels of the nodes are not c.

The above three parameters (τ, e_1, and e_2) can be integrated into a single hypothesis. For example, we can generate a hypothesis to partition the dataset using the following three steps.

Step 1) Generate $\tau \in \{1, 2\}$ to indicate the feature type to use.

Step 2) Generate the dimension index e_1 and e_2 and compute the feature difference $D_i = x_i^{\tau}(e_1) - x_i^{\tau}(e_2)$, $i = 1, 2, \cdots, N$.

Step 3) Split the dataset into two parts based on a threshold on feature difference and obtain a misclassification error.

We generate γ hypotheses independently ($\gamma = 200$ in our experiments) and select the one with the smallest misclassification error. After this, one node will be built and the training set will be partitioned into two parts. For each part, a new node will be further constructed in the same way. This process is repeated until any of the two conditions below is satisfied: 1) the depth of the tree reaches the maximum number or 2) the number of points in the node is smaller than a predefined threshold.

Now, we discuss how to compute $P(\mathbf{C} = c|d_q)$ with a random forest. Suppose we have M trees in a forest and the STIP d_q will fall in one of the leaves in a tree. Assume that, for a tree T_i, the STIP point d_q falls in a leaf with N_i^+ positive samples and N_i^- negative samples. The posterior distribution of d_q can be approximated by the average density of the M nodes in M different trees as

$$P(\mathbf{C} = c|d_q) \approx \frac{1}{M}\sum_{i=1}^{M}\frac{N_i^+}{N_i^+ + N_i^-}. \tag{7}$$

Then, (4) can be replaced with

$$\begin{aligned} S^c(d_q) &= \log P(\mathbf{C} = c|d_q) - \log P(\mathbf{C} = c) \\ &= \log \frac{1}{M}\sum_{i=1}^{M}\frac{N_i^+}{N_i^+ + N_i^-} - \log P(\mathbf{C} = c). \end{aligned} \tag{8}$$

In the training dataset, the numbers of STIP points are different for different action classes. Therefore, it is inaccurate to compute the prior probability $P(\mathbf{C} = c)$ directly from the distribution of training dataset. In our experiments, we introduce the parameter $A = -logP(\mathbf{C} = c)$ and optimize it in the experiments.

The benefits of using the random forest are numerous. First, each tree in the forest is independent of other trees when evaluating $P(\mathbf{C} = c|d_q)$ in (7). The average of them thus reduces the variance of the estimation. Second, random forest is fast to evaluate during the testing stage. The runtime cost for each STIP only depends on the depth of each tree and the number of trees. It is not affected by the number of points in the training data. Hence, it is much faster than LSH-based nearest-neighbor search. In the experiment section, we will show that random forest-based voting approach is over 4000 times faster than the LSH-based approach. Another advantage of random forest compared with LSH is that, when constructing the trees, the label information of x_i can be integrated. Thus, the trees follow the data distribution of the training data. This improves the generalization ability. Finally, the construction of random forest is flexible. Besides the label information, it is easy to combine other types of feature descriptors and spatial information of STIPs.

According to the literature, the work in [6], [46], and [47] also employs tree structures for action recognition. We first consider the differences between [46] and our work. The feature that is employed in [46] is densely sampled while we use the sparse STIP features. Second, the method in [46] votes for the center of the action while our random forest weighs each STIP point so that the nontrivial scale estimation can be partially solved with branch-and-bound search. Third, the votes in [46] are estimated from the frequency view so that it would generate positive votes even for the background. On the contrary, our votes employs the mutual information based measure [see (4)], which is more discriminative thanks to the introduction of negative votes. The trees in [6] are used for indexing and searching nearest neighbors while trees in [47] serve as a codebook. Since we employ random forest to weigh each STIP point, the motivations and implementations are different from those in [6] and [47]. Besides, our work can deal with not only action classification but also action detection, while the methods [6] and [47] are only applicable to action recognition.

After obtaining the individual voting score of each STIP, the spatio-temporal location and scale of the target action will be determined by the branch-and-bound search as described in Section IV.

IV. Action Detection and Localization

The purpose of action detection is to find a subvolume V with the maximum similarity to the predefined action type. Following [9], with each STIP being associated with an individual score $s^c(d)$, our goal is to find the video subvolume with the maximum score

$$V^* = argmax_{V \subset \mathcal{V}} f(V) \tag{9}$$

where $V = [T, B] \times [L, R] \times [S, E]$ is a video subvolume, where L, R, T, B, S and E are the left, right, top, bottom, start, and end positions of V; $f(V) = \sum_{d \in V} s^c(d)$ and $\mathcal{V}$ is the whole video space. A subvolume V is said to be *maximal* if there does not exist any other subvolume V' such that $f(V') > f(V)$ and $V' \cap V \neq \emptyset$. The action detection problem is to find all the maximal subvolumes whose scores are above a certain threshold.

A spatio-temporal branch-and-bound algorithm was proposed in [9] to solve the single subvolume search problem. Instead of performing a branch-and-bound search directly in the 6-D parameter space Λ, the method performs a branch-and-bound search in the 4-D spatial parameter space. In other words, it finds the spatial window W^* that maximizes the following function:

$$F(W) = \max_{T \subseteq \mathbb{T}} f(W \times T) \tag{10}$$

where $W = [T, B] \times [L, R]$ is the spatial window, $T = [S, E]$ is the temporal segment, and $\mathbb{T} = [0, t - 1]$.

One advantage of separating the parameter space is that the worst-case complexity is reduced from $O(m^2n^2t^2)$ to $O(m^2n^2t)$. The complexity is linear in t, which is usually the largest of the three dimensions. For this reason, it is efficient in processing long videos, but, when the spatial resolution of the video increases, the complexity goes up quickly. The method in [9] was tested on videos with low resolution (160 $\times$ 120). In this paper, we are interested in higher resolution videos (320 $\times$ 240 or higher). We found that, for videos taken under challenging lighting conditions with crowded background such as those in the publicly available MSR Action dataset II,[2] the action detection rates on 320 $\times$ 240 resolution videos are much better than those on 160 $\times$ 120. Unfortunately, the subvolume

[2]The MSR action dataset II is available at http://research.microsoft.com/en-us/um/people/zliu/ActionRecoRsrc/default.htm.

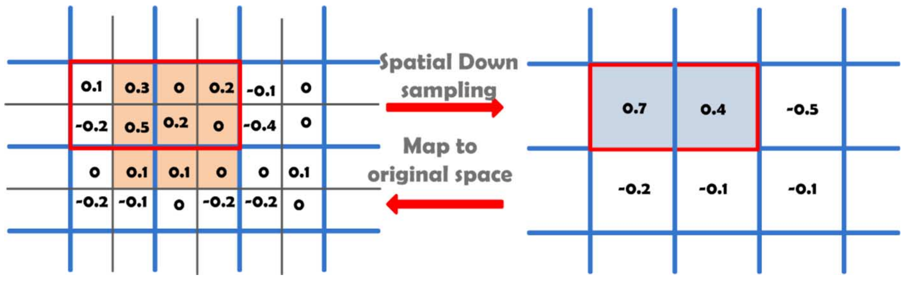


Fig. 2. Approximation of the spatial down-sampling. Left: score image in the original resolution. Right: down-sampled score image. Every four small pixels in a cell from the original resolution sum up to one score in the low resolution, for example, the value in the top-left pixel from the right figure $0.5 = 0.1 + 0.3 - 0.4 + 0.5$. We notice that the optimal solution found in the down-sampled video space is worse than that in the original space ($f^s(\tilde{V}^*) = 1.1 < f(V^*) = 1.4$).

search for 320 × 240 videos is much slower. For example, [9] takes 20 h to search the MSR Action dataset II which consists of 54 video sequences each 1 min long with 320 × 240 resolution.

Moreover, in [9], the multi-instance detection problem was converted to a series of single subvolume search problem. They first find the optimal subvolume V_1 such that $f(V_1) = max_V f(V)$. After that, it sets the scores of all of the points in V_1 to 0, and finds the optimal subvolume V_2, and so on. To further speed up the search process during the branch-and-bound iterations, a heuristic was used in [12]. If a candidate window $\mathbb{W}$ with a score larger than the detection threshold is found, the subsequent searches are limited to the subwindows contained in $\mathbb{W}$. It guarantees that it will find a valid detection, but the detected subvolume is not guaranteed to be optimal.

In the next two subsections, we present two techniques to speed up the subvolume search algorithm. The combination of the two techniques allow us to perform subvolume search on 320 × 240 videos in real time.

A. Spatial Down-Sampling

To handle high-resolution videos, the technique is to spatially down-sample the video space by a factor s before the branch-and-bound search. Note that the interest point detection, descriptor extraction, and the scores are all done in the original video sequence.

For a video volume $\mathcal{V}$ of size $m \times n \times t$, the size of the down-sampled volume $\mathcal{V}^s$ with scale factor s is $(m/s) \times (n/s) \times t$. For any point $(i,j,k) \in \mathcal{V}^s$ where $i \in [0,(m/s)-1]$, $j \in [(n/s)-1]$, and $k \in [0,t-1]$, its score is defined as the sum of the scores of the $s \times s$ points in $\mathcal{V}$, that is, $f^s(i,j,k)$ is defined as

$$f^s(i,j,k) = \sum_{x=0}^{s-1}\sum_{y=0}^{s-1} f(s*i+x, s*j+y, k). \tag{11}$$

Given any subvolume $V^s = [L,R] \times [T,B] \times [S,E] \subset \mathcal{V}^s$, where L, R, T, B, S, and E are the left, right, top, bottom, start, and end positions of V^s, respectively, denote $\xi(V^s)$ as its corresponding subvolume in original video $\mathcal{V}$, that is,

$$\begin{aligned}\xi(V^s) = [s*L, s*(R+1)-1]\\ \times [s*T, s*(B+1)-1] \times [S,E].\end{aligned} \tag{12}$$

As they are the same subvolume, it is easy to see that

$$f^s(V^s) = f\left(\xi(V^s)\right). \tag{13}$$

A subvolume $V = [X_1, X_2] \times [Y_1, Y_2] \times [T_1, T_2] \subset \mathcal{V}$ is called an *s-aligned* subvolume if X_1 and Y_1 are multiples of s and the width $X_2 - X_1 + 1$ and height $Y_2 - Y_1 + 1$ are also multiples of s. Equation (12) provides a one-to-one mapping between the volumes in $\mathcal{V}^s$ and the s-aligned subvolumes in $\mathcal{V}$.

Instead of searching the original video space, we can search the down-sampled video space $\mathcal{V}^s$ of a much smaller size $(m/s) \times (n/s) \times t$. However, as the down-sampling process also introduces the approximation errors, it affects the search results. In general, for any $V^s \subset \mathcal{V}^s$, there exists a $V = \xi(V^s) \subset \mathcal{V}$. It thus shows that the maximum subvolume found in the down-sampled space is at most as good as the one found in the original space

$$\max_{V^s \subset \mathcal{V}^s} f^s(V^s) \le \max_{V \subset \mathcal{V}} f(V). \tag{14}$$

We illustrate a concrete example in Fig. 2. For simplicity, in Fig. 2, we choose the down-sampling factor $s = 2$ and discuss the problem in the $2D$ space (only one frame is considered). The left figure shows the original video space, and its down-sampled version is in the right figure. Each pixel is associated with a voting score. The orange rectangle highlights the optimal solution in the original video space, namely the bounding box of the highest total sum. After the down-sampling, the grey rectangle is the detection result in the down-sampled video. By mapping it back to the original space, we obtain an approximate solution highlighted by the red rectangle. It overlaps with the optimal solution in the original space, but the total sum is slightly less. To further quantify the approximation error, we derive the upper bound of the error caused by the down-sampling, as explained in Theorem 1.

Theorem 1: Bound of the Approximation Error: Let V^* denote the optimal subvolume in $\mathcal{V}$, that is, $f(V^*) = \max_{V \subset \mathcal{V}} f(V)$. Assume $V^* = [x_1, x_1 + w - 1] \times [y_1, y_1 + h - 1] \times [t_1, t_2]$ where w and h are the width and height of V^*, respectively, and further assume the total score of a subvolume is on average proportional to its size. Then, there exists an s-aligned subvolume $\tilde{V}$ satisfying

$$f(\tilde{V}) \ge \left(1 - \frac{s*h + s*w + s^2}{wh}\right) f(V^*). \tag{15}$$

The proof of this theorem is given in the Appendix.

Let $\tilde{V}^* = \mathrm{argmax}_{V \in \mathcal{V}^s f^s(V)}$ denote the optimal subvolume in $\mathcal{V}^s$. Based on (15), we have

$$f^s(\tilde{V}^*) \ge \left(1 - \frac{s*h + s*w + s^2}{wh}\right) f(V^*). \tag{16}$$

As an example, suppose the spatial dimension of V is 320 $\times$ 240 and the scale factor $s = 8$. The spatial dimension of the down-sampled volume is 40 $\times$ 30. If we assume the window size of the optimal subvolume V^* is 64 $\times$ 64, then the average relative error is at most

$$\frac{s*h+s*w+s^2}{wh} = \frac{8*64+8*64+8^2}{64^2} \approx 25\%. \quad (17)$$

We have run numerical experiments to measure the relative error of the optimal solutions in the down-sampled volumes. We used 30 video sequences of resolution 320 $\times$ 240. There are three action types. For each video sequence and each action type, we obtain a 3-D volume of scores as defined in (8). We choose $s = 8$, and down-sample each 3-D volume to spatial resolution of 40 $\times$ 30. There are 113 actions in total. For each action, we compute its corresponding down-sampled subvolume and evaluate the relative error which is the score difference divided by the original action score. The mean is 23%, and the standard deviation is 26%. We can see that the numerical experiments are consistent with the theoretical analysis.

B. Top-K Search Algorithm

The multi-instance search algorithm in [9] repeatedly applies the single-instance algorithm many times until some stop criteria is met. In practice, there are typically two different stop conditions that can be used. The first is to stop after k, iterations where k is a user-specified integer. The second is to stop when the detection score is smaller than a user-specified detection threshold λ. In either case, suppose the number of detected instances is k, then the worst case complexity of the algorithm is $O(kn^2m^2t)$.

We notice that, in the 1-D case, Brodal and Jorgensen [2] developed an algorithm that finds the top-K subarrays in $O(n+k)$ time. This is much more efficient than repeatedly applying the single-instance algorithm k times, which has the complexity $O(kn)$. In a 3-D case, we would also like to have an algorithm that is more efficient than simply applying the single-instance algorithm k times. We consider two different variants corresponding to the two stop criteria. The first, called λ search, can be applied when we are interested in finding all of the subvolumes above a user-specified threshold λ. The second, called top-K search, can be applied when we are interested in finding the top-K subvolumes.

1) λ Search: Here, we describe an algorithm that finds all of the subvolumes with scores larger than a user-specified threshold λ. The pseudo-code of the algorithm is shown in Algorithm 1. Following the notation in [9], we use $\mathbb{W}$ to denote a collection of spatial windows, which is defined by four intervals that specify the parameter ranges for the left, right, top, and bottom positions, respectively. Given any set of windows $\mathbb{W}$, we use $\hat{F}(\mathbb{W})$ to denote its upper bound which is estimated in the same way as in [9] and [48]. We use $W_{\max}$ to denote the largest window among all of the windows in $\mathbb{W}$. Initially, $\mathbb{W}^*$ is equal to the set of all of the possible windows on the image and $F*$ is the corresponding upper bound, as in Line 5 of Algorithm 1. From Lines 6–19, we split and store the results if the top state $\mathbb{W}$ is over a threshold λ and iterate this process. From Lines 20–22, we have a subvolume (V^*) detected. The whole process iterates until the score for the detected subvolume is below the threshold.

Algorithm 1 λ search.

```
1: Initialize $P$ as empty priority queue
2: set $\mathbb{W} = [T,B,L,R] = [0,m] \times [0,m] \times [0,n] \times [0,n]$
3: push($\mathbb{W}, \hat{F}(\mathbb{W})$) into $P$
4: **repeat**
5:   Initialize current best solution $F^*$, $W^*$
6:   **repeat**
7:     retrieve top state $\mathbb{W}$ from P based on $\hat{F}(\mathbb{W})$
8:     **if** $\hat{F}(\mathbb{W}) > \lambda$ **then**
9:       split $\mathbb{W}$ into $\mathbb{W}^1 \cup \mathbb{W}^2$
10:      **if** $\hat{F}(\mathbb{W}^1) > \lambda$ **then**
11:        push $(\mathbb{W}^1, \hat{F}(\mathbb{W}^1))$ into $P$
12:        update current best solution $\{W^*, F^*\}$
13:      **end if**
14:      **if** $\hat{F}(\mathbb{W}^2) > \lambda$ **then**
15:        push $(\mathbb{W}^2, \hat{F}(\mathbb{W}^2))$ into $P$
16:        update current best solution $\{W^*, F^*\}$
17:      **end if**
18:    **end if**
19:  **until** $\hat{F}(\mathbb{W}) \le F^*$
20:  $T^* = argmax_{T\in[0,t]} f(W^*, T)$;
21:  add $V^* = [W^*, T^*]$ to the list of detected subvolumes.
22:  for each point $(i,j,k) \in V^*$, set $f(i,j,k) = 0$.
23: **until** $\hat{F}(\mathbb{W}) \le \lambda$
```

In terms of the worst case complexity, the number of branches of this algorithm is no larger than $O(n^2m^2)$, since the algorithm does not restart the priority queue P. Each time it branches, the algorithm has to compute the upper bound whose complexity is $O(t)$. Therefore, the worst complexity involved in branch and bound is the same as in [9]: $O(n^2m^2t)$. In addition, each time it detects a subvolume, the algorithm has to update the scores of the video volume which has complexity $O(nmt)$. If there are k detected subvolumes, the complexity for updating the scores is $O(kmnt)$. Overall, the worst case complexity of this algorithm is $O(n^2m^2t) + O(kmnt)$. When k is large, this is much better than $O(kn^2m^2t)$.

2) Top-K Search: Here, we describe how to modify Algorithm 1 for the case when we are interested in finding the top-K actions, and we assume we do not know the threshold λ.

The pseudo-code of the algorithm is shown in Algorithm 2. The algorithm is similar to Algorithm 1. In Line 6, $(\{W_i^*, F_i^*\})_{i=c\ldots k}$ are set as all of the possible windows on the image and its upper bound score, respectively. From Line 6–20, we split and store the results if the top state $\mathbb{W}$ is over the Kth top score and iterate this process. From Lines 21–24, we have a subvolume (V^*) detected. The whole process iterates until

K subvolumes are detected. There are four major differences. First, instead of maintaining a single current best solution, it maintains k-best current solutions. Second, it replaces the criteria $\hat{F}(\mathbb{W}) > \lambda$ with $\hat{F}(\mathbb{W}) > F_k^*$ to determine whether we need to insert $\mathbb{W}^1$ or $\mathbb{W}^2$ into the queue P. Third, it replaces the inner-loop stop criteria $\hat{F}(\mathbb{W}) \le F^*$ with $\hat{F}(\mathbb{W}) \le F_c^*$. Finally, the outer-loop stop criteria $\hat{F}(\mathbb{W}) \le \lambda$ is replaced with $c > k$. In this algorithm, the number of outer loops is k. Thus, the worst case complexity is also $O(n^2m^2t) + O(kmnt)$.

Algorithm 2 Top-K Search.

```
1: Initialize $P$ as empty priority queue
2: set $\mathbb{W} = [T, B, L, R] = [0, m] \times [0, m] \times [0, n] \times [0, n]$
3: push($\mathbb{W}, \hat{F}(\mathbb{W})$) into $P$
4: c $= 1$
5: **repeat**
6:   Initialize $(\{W_i^*, F_i^*\})_{i=c\ldots k}$ where $F_k^* \le \ldots \le F_c^*$
7:   **repeat**
8:     retrieve top state $\mathbb{W}$ from P based on $\hat{F}(\mathbb{W})$
9:     **if** $\hat{F}(\mathbb{W}) > F_k^*$ **then**
10:      split $\mathbb{W}$ into $\mathbb{W}^1 \cup \mathbb{W}^2$
11:      **if** $\hat{F}(\mathbb{W}^1) > F_k^*$ **then**
12:        push $(\mathbb{W}^1, \hat{F}(\mathbb{W}^1))$ into $P$
13:        update $(\{W_i^*, F_i^*\})_{i=c\ldots k}$
14:      **end if**
15:      **if** $\hat{F}(\mathbb{W}^2) > F_k^*$ **then**
16:        push $(\mathbb{W}^2, \hat{F}(\mathbb{W}^2))$ into $P$
17:        update $(\{W_i^*, F_i^*\})_{i=c\ldots k}$
18:      **end if**
19:    **end if**
20:  **until** $\hat{F}(\mathbb{W}) \le F_c^*$
21:  $T^* = argmax_{T\in[0,t]} f(W^*, T)$;
22:  output $V_c^* = [W^*, T^*]$ as the $c$-th detected subvolume
23:  for each point $(i, j, k) \in V_c^*$, set $f(i, j, k) = 0$.
24:  $c = c + 1$
25:**until** $c > k$
```

V. Experiments

A. Action Classification

To evaluate our proposed random forest based approach for multiclass action classification, we test on the benchmark KTH dataset. The experiment setup is the same as in [1] and [9], where clips from 16 persons are used for training, and the other nine persons are used for testing. The confusion matrix is listed in Table I. We also compare our results with the state-of-the-art results in Table II. With the same input features, our method performs as well as the method using support vector machine for classification [4]. Although our performance is slightly worse than the nearest-neighbor-based classification in [9], as will be shown later, our approach is significantly faster as it avoids the nearest neighbor search.

TABLE I
Confusion Matrix for KTH Action Dataset. The Total Accuracy is 91.8%

	clap	wave	box	run	jog	walk
clap	137	1	6	0	0	0
wave	7	137	0	0	0	0
box	0	0	144	0	0	0
run	0	0	0	95	47	2
jog	0	0	0	4	136	4
walk	0	0	0	0	0	144

B. Action Detection

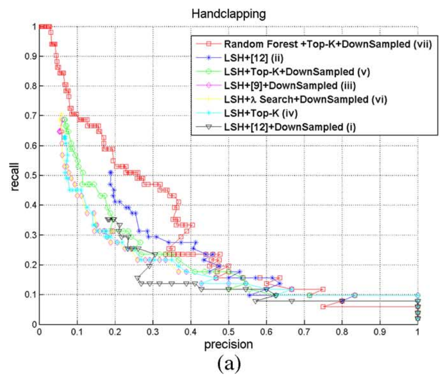
To evaluate our multiclass action detection and localization, we perform cross-dataset training and testing. We first build a random forest using the KTH dataset (with 16 persons in the training part) and then test on a challenging dataset (MSRII) of 54 video sequences where each video consists of several actions performed by different people in a crowded environment. Each video is approximately one minute long. The videos contain three different types of actions: handwaving, handclapping, and boxing. Some videos contain different people performing different actions simultaneously. There are also instances where a person performs two different actions consecutively.

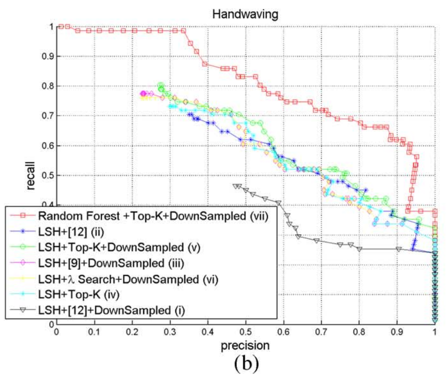
For all of our experiments, we have fixed $K = 3$, $\lambda = 3.0$. Moreover, unless explicitly mentioned, we down-sample the score volume to 40 $\times$ 30 pixels.

Fig. 3 compares the precision-recall for the following methods (the original videos are of high resolution 320 $\times$ 240).

1) Accelerated spatio-temporal branch-and-bound search (ASTBB) of [12] in low-resolution score volume (frame size 40 $\times$ 30).
2) ASTBB of [12] in 320 $\times$ 240 videos.
3) Multiround branch-and-bound search of [9] in low-resolution score volume (frame size 40 $\times$ 30).
4) Top-K search at original size 320 $\times$ 240.
5) Top-K search at down-sampled score volume (size 40 $\times$ 30).
6) λ search at down-sampled score volume (size 40 $\times$ 30).
7) Random forest-based weighting followed by top-K search at down-sampled score volume (size 40 $\times$ 30).

Except for 7), which uses our random forest based voting score, the other methods apply the LSH-based nearest-neighbor voting score as in [9]. The parameter $A = -\log P(C = c)$ in (8) for method 7) is set to 2.1, 1.7, and 0.9 for handclapping, handwaving, and boxing, respectively. Also, we use the walking actions from KTH as the negative dataset when constructing forests. For the purpose of generating precision-recall curves, we modified the outer-loop stop criteria (line 25, Algorithm 2) to repeat until $\hat{F}(\mathbb{W}) \le \lambda$, where λ is a small threshold. In this way, it outputs more than K subvolumes, which is necessary for plotting the precision-recall curve. Some sample detection results obtained by our approach 7) are shown in Fig. 5. To demonstrate the capability of handling nonstationary actions, we show a walking detection result at the bottom row. The detection is done by using KTH walking as the positive training





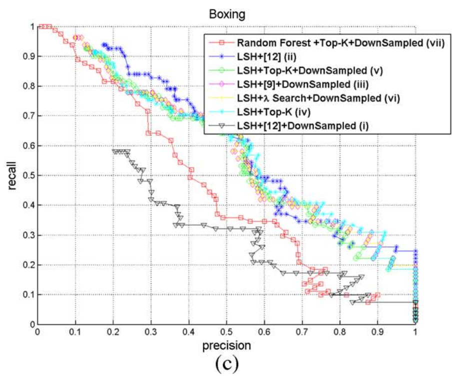


Fig. 3. Precision-recall curves for action detections with different methods. (a) Handclapping. (b) Handwaving. (c) Boxing.

TABLE II
COMPARISON OF DIFFERENT REPORTED RESULTS ON KTH DATASET

Method	Mean accuracy
Our method	91.8%
Yuan et al's [9]	93.3%
Reddy et al's [6]	90.3%
Laptev et al's [4]	91.8%

TABLE III
TIME CONSUMED FOR VOTING ONE STIP AND ONE VIDEO SEQUENCE (FOR EXAMPLE, 10000 STIP POINTS). ONLY CPU TIME IS CONSIDERED

Method	Voting Time (ms)	One sequence (s)
LSH	18.667±8.4105	186.67
Random Forest	0.0042±0.0032	0.042

data while the KTH handwaving, handclapping, and boxing are used as the negative training data.

The measurement of precision and recall is the same as what is described in [9]. For the computation of the precision, we consider a true detection if: $\mathrm{Volume}(V^* \cap G)/\mathrm{Volume}(G) > 1/8$, where G is the annotated ground truth subvolume, and V^* is the detected subvolume. On the other side, for the computation of the recall we consider a hit if: $\mathrm{Volume}(V^* \cap G)/\mathrm{Volume}(V^*) > 1/8$.

We first compare the results based on LSH voting approaches. Fig. 3 lists the Precision-Recall curves for the three different action classes, respectively. Fig. 4 shows the average PR curve for the three actions. The average PR curve is computed by averaging precision and recall results among the three action types while adjusting a threshold. This can give a general idea of the overall performance for different algorithms. From the precision-recall curves, we can see that although the accelerated search of [12] provides excellent results in high resolution videos, its performance on down-sampled low resolution videos is poor compared with other search schemes. Moreover, all the methods applied to the high resolution videos provide similar performance. In particular, the methods of top-K search with branch-and-bound search at down-sampled size (v) and λ search with branch-and-bound search at down-sampled size (vi) are among the best ones. These results justify our proposed λ search and top-K search algorithms. Although the branch-and-bound is performed in the down-sampled size videos, it still provides good performance. However, the search speed is much faster. To compare the performance of action detection between LSH and random forest, (v) and (vii) are two search schemes with the same environment but different voting approaches. Random

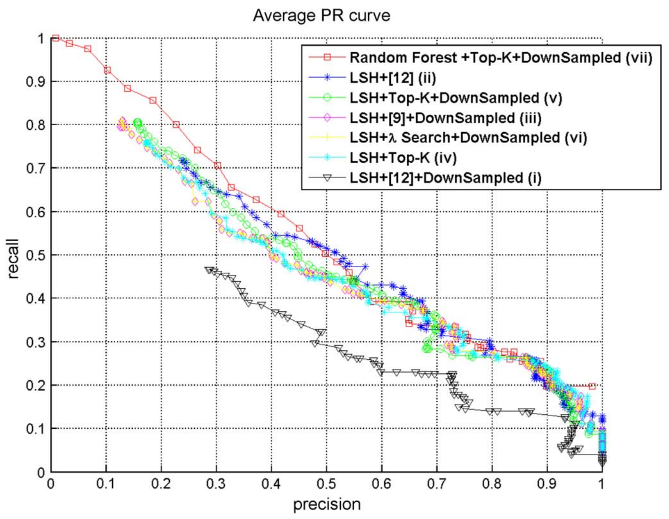


Fig. 4. Comparisons of average precision-recall curves.

TABLE IV
TIME CONSUMED FOR EACH METHOD TO SEARCH ACTIONS IN THE 54 VIDEOS

Method	Running Time
Low resolution (40×30) [12]	40 mins
High resolution (320×240) [12]	20 hours
Down-sampled B&B (40×30)	10 hours
λ search + Down-sampled B&B (40×30)	1 hour 20 mins
Top-K + Down-sampled B&B (80×60)	6 hours
Top-K + Down-sampled B&B (40×30)	26 mins

forest (vii) is superior to LSH (v) in handwaving but poorer in boxing. Since the boxing action is highly biased in KTH dataset (much more boxing actions are performed from right to left), it reduces the discriminative ability of the trees. For LSH, however, because it searches only one nearest positive and negative in the neighborhood, the effect of such bias can almost be ignored.

C. Computational Cost

The feature extraction step is performed with publicly available code in [1]. Although their code may not be very fast, there are faster implementations available. Therefore, the computation time for feature extraction is not considered in this paper. We suppose that all of the STIP points are already extracted and stored in the memory. Then, the computational time of our algorithms is dominated by two operations, computing the score for

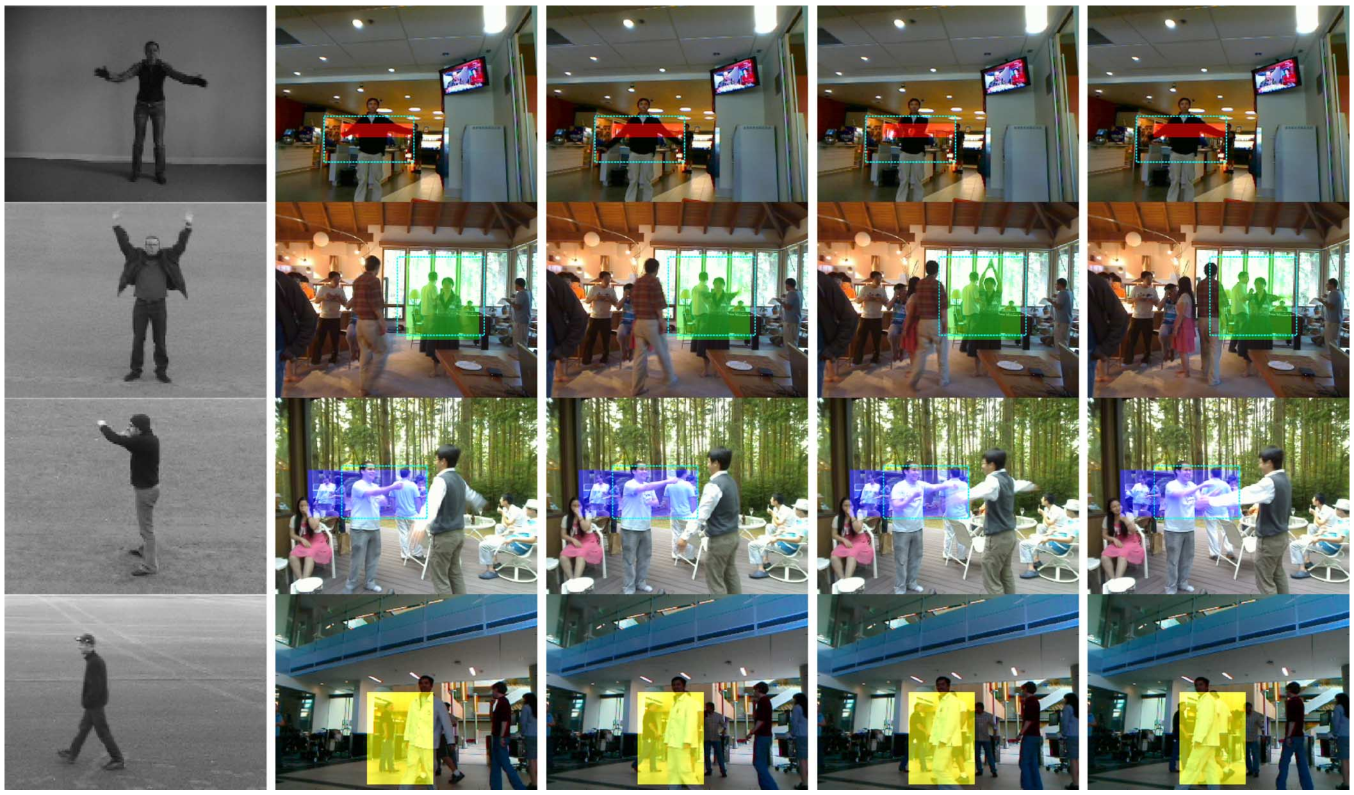


Fig. 5. Detection results (Random Forest+Top-K) of handclapping (first row), handwaving (second row), boxing (third row), and walking (fourth row) are listed in columns 2–5 with red, green, blue, and yellow (online version) to show the bounding boxes, respectively. The cyan dashed regions are the ground truths. The first column shows sample images from training set.

TABLE V
COMPARISON OF TOTAL TIME COST FOR ACTION DETECTION. ONLY CPU TIME IS CONSIDERED

Method	Voting Time (mins)	Search Time (mins)	Total Computation Time (mins)
LSH+B&B [12]	271	1200	1471
LSH+Top-K (our algorithm)	271	26	297
Random Forest+Top-K (our algorithm)	0.62	26	26.62

each STIP and branch-and-bound search. For the first part, LSH takes, on average, 18.667 ms per STIP point while random forest only takes 0.0042 ms. To deal with a video clip with 10 000 STIPs, it will take around 186.67 s for LSH but only 42 ms for random forest, that is, random forest-based approach is 4000 times faster than LSH-based approach.

Table IV shows the time consumed for the search part. All of the algorithms are implemented using C++, performed on a single PC of dual-core and 4-G main memory: 1) accelerated λ search of [12] in low-resolution videos (frame size 40×30); 2) accelerated λ search of [12] in high-resolution videos; 3) multiround branch-and-bound search of [9] in low-resolution videos (frame size 40×30); 4) λ search, with branch-and-bound search at down-sampled size 40×30; and 5) top-K search, with branch-and-bound search at down-sampled size 80×60, (f) top-K search, with branch-and-bound search at down-sampled size 40×30.

Table IV shows that, although the method of [12] works well for low-resolution videos, the search speed becomes much slower for high-resolution videos. Moreover, as shown in Fig. 4, when performing on the down-sampled score volumes, the heuristic method of [12] (curve (i)) is a lot worse than the other methods. This is an indication that it is not a good idea to perform heuristic search on down-sampled score volumes. In comparison, λ search provides much better search quality. Among all the search schemes, the fastest method is the top-K search with branch-and-bound at down-sampled score volume of 40×30. It takes only 26 min to process the 54 sequences whose total length is about one hour in total.

Finally, we compare LSH and random forest in terms of total computation time in Table V, including the runtime cost for computing scores and the runtime cost for top-K search. For the previous method [12], it takes at least 1471 min to search all the actions for 54 videos in MSRII. In contrast, the total computation time of our proposed algorithm is 26.62 min.

VI. CONCLUSION

We have developed a new system for the spatio-temporal localization of human actions in video sequences. The system improves upon the state of the art in two aspects. First, we proposed a random forest-based voting technique to compute the scores of the interest points, which achieves a multiple orders-of-magnitude speed-up compared with the nearest-neighbor-based scoring scheme. Second, we proposed a top-k search technique which detects multiple action instances simultaneously with a single round of branch-and-bound search. To reduce the computational complexity of searching higher resolution videos, we performed a subvolume search on the down-sampled score volumes. We have presented experiment results on challenging videos with crowded background. The results showed that our proposed system is robust to dynamic and cluttered background and is able to perform faster-than real-time action detection on high-resolution videos.

Appendix

Here, we prove Theorem 1. Let V^* denote the optimal subvolume in $\mathcal{V}$, that is, $f(V^*) = max_{V \subset \mathcal{V}} f(V)$. Assume $V^* = [x_1, x_1+w-1] \times [y_1, y_1+h-1] \times [t_1, t_2]$, where w and h are the width and height of V^*, respectively. Let $|V|$ denote the number of voxels in V. It can be shown that there exists an s-aligned subvolume $\tilde{V} = [\tilde{x}_1, \tilde{x}_1 + \tilde{w} - 1] \times [\tilde{y}_1, \tilde{y}_1 + \tilde{h} - 1] \times [t_1, t_2]$ such that

$$\left|(V^* \setminus \tilde{V}) \cup (\tilde{V} \setminus V^*)\right| \leq (s*h + s*w + s^2)(t_2 - t_1). \quad (18)$$

Therefore

$$\frac{\left|(V^* \setminus \tilde{V}) \cup (\tilde{V} \setminus V^*)\right|}{|V^*|} \leq \frac{s*h + s*w + s^2}{wh}. \quad (19)$$

If we assume the total score of a subvolume is on average proportional to its size, then

$$\frac{f\left((V^* \setminus \tilde{V}) \cup (\tilde{V} \setminus V^*)\right)}{f(V^*)} \leq \frac{s*h + s*w + s^2}{wh}. \quad (20)$$

Therefore

$$\frac{f(V^*) - f(\tilde{V})}{f(V^*)} \leq \frac{s*h + s*w + s^2}{wh}. \quad (21)$$

After a rearrangement of the items, we have

$$f(\tilde{V}) \geq \left(1 - \frac{s*h + s*w + s^2}{wh}\right) f(V^*). \quad (22)$$

References

[1] I. Laptev, “On space-time interest points,” *Int. J. Comput. Vis.*, vol. 64, no. 2–3, pp. 107–123, 2005.
[2] G. Brodal and A. Jørgensen, “A linear time algorithm for the k maximal sums problem,” *Math. Foundations Comput. Sci.*, pp. 442–453, 2007.
[3] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: A local SVM approach,” in *Proc. IEEE Conf. Pattern Recognit.*, 2004.
[4] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008.
[5] J. Gall and V. Lempitsky, “Class-specific Hough forests for object detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009.
[6] K. K. Reddy, J. Liu, and M. Shah, “Incremental action recognition using feature-tree,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009.
[7] E. Shechtman and M. Irani, “Space-time behavior based correlation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005.
[8] Y. Ke, R. Sukthankar, and M. Hebert, “Event detection in crowded videos,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2007.
[9] J. Yuan, Z. Liu, and Y. Wu, “Discriminative subvolume search for efficient action detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009.
[10] Y. Hu, L. Cao, F. Lv, S. Yan, Y. Gong, and T. S. Huang, “Action detection in complex scenes with spatial and temporal ambiguities,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009.
[11] A. F. Bobick and J. W. Davis, “The recognition of human movement using temporal templates,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 3, pp. 257–267, Mar. 2001.
[12] J. Yuan, Z. Liu, Y. Wu, and Z. Zhang, “Speeding up spatio-temporal sliding-window search for efficient event detection in crowded videos,” in *Proc. ACM Multimedia Workshop on Events in Multimedia*, 2009.
[13] M. D. Rodriguez, J. Ahmed, and M. Shah, “Action mach a spatio-temporal maximum average correlation height filter for action recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008.
[14] E. Boyer, D. Weinland, and R. Ronfard, “Free viewpoint action recognition using motion history volumes,” *Comput. Vis. Image Understanding*, vol. 104, no. 2–3, pp. 207–229, 2006.
[15] Y. Ke, R. Sukthankar, and M. Hebert, “Efficient visual event detection using volumetric features,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2005.
[16] M. Yang, F. Lv, W. Xu, K. Yu, and Y. Gong, “Human action detection by boosting efficient motion features,” in *Proc. IEEE Workshop Video-oriented Object and Event Classification in Conjunction With ICCV*, Kyoto, Japan, Sep. 29–Oct. 2 2009.
[17] Z. Lin, Z. Jiang, and L. S. Davis, “Recognizing actions by shape-motion prototype trees,” in *Proc. IEEE Intl. Conf. Comput. Vis.*, 2009.
[18] H. Jiang, M. S. Drew, and Z. N. Li, “Action detection in cluttered video with successive convex matching,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 1, pp. 50–64, Jan. 2010.
[19] L. Cao, Z. Liu, and T. S. Huang, “Cross-dataset action recognition,” in *Proc. IEEE Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2010.
[20] A. Norbert, Z. Liu, and J. Yuan, “Efficient search of top-K video subvolumes for multi-instance action detection,” in *Proc. IEEE Conf. Multimedia Expo (ICME)*, 2010.
[21] K. G. Derpanis, M. Sizintsev, K. Cannons, and R. P. Wildes, “Efficient action spotting based on a spacetime oriented structure representation,” in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2010.
[22] C. H. Lampert, M. B. Blaschko, and T. Hofmann, “Efficient subwindow search: A branch and bound framework for object localization,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2129–2142, Dec. 2009.
[23] C. H. Lampert, “Detecting objects in large image collections and videos by efficient subimage retrieval,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009.
[24] L. Breiman, “Random forests,” *Mach. Learning*, vol. 45, pp. 5–32, 2001.
[25] A. Bosch, A. Zisserman, and X. Munoz, “Image classification using random forests and ferns,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2007.
[26] V. Lepetit, P. Lagger, and P. Fua, “Randomized trees for real-time keypoint recognition,” in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2005.
[27] F. Schroff, A. Criminisi, and A. Zisserman, “Object class segmentation using random forests,” in *Proc. Brit. Mach. Vis. Conf.*, 2008.
[28] P. Wang, G. D. Abowd, and J. M. Rehg, “Quasi-periodic event analysis for social game retrieval,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009.
[29] K. Prabhakar, S. Oh, P. Wang, G. D. Abowd, and J. M. Rehg, “Temporal causality for the analysis of visual events,” in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2010.
[30] J. Liu, J. Luo, and M. Shah, “Recognizing realistic actions from videos “in the wild”,” in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2009.
[31] R. Messing, C. Pal, and H. Kautz, “Activity recognition using the velocity histories of tracked keypoints,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009.

[32] L. Duan, D. Xu, I. W. Tsang, and J. Luo, "Visual event recognition in videos by learning from web data," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2010.
[33] A. Kovashka and K. Grauman, "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2010.
[34] J. Niebles, C. W. Chen, and F.-F. Li, "Modeling temporal structure of decomposable motion segments for activity classification," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010.
[35] H. J. Seo and P. Milanfar, "Detection of human actions from a single example," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2009.
[36] K. Mikolajczyk and H. Uemura, "Action recognition with motion-appearance vocabulary forest," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2008.
[37] Y. Yacoob and M. J. Black, "Parameterized modeling and recognition of activities," *Proc. Comput. Vis. Image Understanding Conf.*, vol. 73, pp. 232–247, 1999.
[38] D. Ramanan and D. A. Forsyth, "Automatic annotation of everyday movements," in *Proc. Neural Inf. Process. Syst. Conf.*, 2003.
[39] T. K. Kim and R. Cipolla, "Canonical correlation analysis of video volume tensors for action categorization and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 8, pp. 1415–1428, Aug. 2008.
[40] L. Cao, Y. L. Tian, Z. Liu, B. Yao, Z. Zhang, and T. S. Huang, "Action detection using multiple spatio-temporal interest point features," in *Proc. IEEE Conf. Multimedia Expo*, 2010.
[41] H. J. Seo and P. Milanfar, "Action recognition from one example," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 867–882, May 2010.
[42] Z. Li, Y. Fu, S. Yan, and T. S. Huang, "Real-time human action recognition by luminance field trajectory analysis," in *Proc. ACM Int. Conf. Multimedia*, 2008.
[43] G. Zhu, M. Yang, K. Yu, W. Xu, and Y. Gong, "Detecting video events based on action recognition in complex scenes using spatio-temporal descriptor," in *Proc. ACM Int. Conf. Multimedia*, Oct. 19–24 2009, pp. 165–174.
[44] M. Breitenbach, R. Nielsen, and G. Z. Grudic, "Probabilistic random forests: Predicting data point specific misclassification probabilities," Univ. of Colorado at Boulder, Tech. Rep. CU-CS-954-03, 2003.
[45] [Online]. Available: http://en.wikipedia.org/wiki/Pointwise_mutual_information
[46] A. Yao, J. Gall, and L. Van Gool, "A hough transform-based voting framework for action recognition," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2010.
[47] T. H. Yu, T. K. Kim, and R. Cipolla, "Real-time action recognition by spatiotemporal semantic and structural forest," in *Proc. BMVC*, 2010.
[48] J. Yuan, Z. Liu, and Y. Wu, "Discriminative video pattern search for efficient action detection," *IEEE Trans. Pattern Anal. Mach. Intell.*.
[49] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2009.

Gang Yu (S'11) received the M.S. degree in computer science from Shanghai Jiao Tong University, Shanghai, China. He is currently working toward the Ph.D. degree at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

His research interests include computer vision and machine learning.

Norberto A. Goussies received the B.S. and M.S. degrees in computer science from Universidad de Buenos Aires, Buenos Aires, Argentina, and the M.S. degree in mathematics, vision, and learning from l'Ecole Normale Superieure de Cachan, Cachan, France. He is currently working toward the Ph.D. degree at the Image Processing Group, Universidad de Buenos Aires.

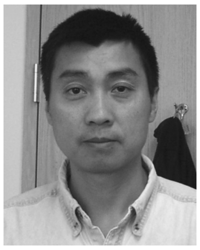
His research interests include computer vision, machine learning, and optimization.

Mr. Goussies was the recipient of several scholarships, including the CONICET Scholarship and the ENS Cachan International Scholarship.

Junsong Yuan (M'08) received the B.Eng. degree in communication engineering from Huazhong University of Science and Technology, Wuhan, China, the M.Eng. degree in electrical engineering from the National University of Singapore, and the Ph.D. degree in electrical engineering from Northwestern University, Evanston, IL, in 2009.

He joined Nanyang Technological University, Singapore, as a Nanyang Assistant Professor in September 2009. He has been a Research Intern with the Communication and Collaboration Systems Group, Microsoft Research, Redmond, WA, Kodak Research Laboratories, Rochester, NY, and Motorola Applied Research Center, Schaumburg, IL. From 2003 to 2004, he was a Research Scholar with the Institute for Infocomm Research, Singapore. He has filed three U.S. patents. He currently serves as an editor for the *KSII Transactions on Internet and Information Systems*. His current research interests include computer vision, image and video data mining and content analysis, machine learning, and multimedia search.

Dr. Yuan is a member of the Association for Computing Machinery. He was the recipient of the Outstanding Ph.D. Thesis award from the Electrical Engineering and Computer Science Department of Northwestern University and the Doctoral Spotlight Award from the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09). He was also a recipient of the Nanyang Assistant Professorship from Nanyang Technological University. In 2001, he was the recipient of the National Outstanding Student and Hu Chunan Scholarship by the Ministry of Education in China.



Zicheng Liu (SM'05) received the B.S. degree in mathematics from Huazhong Normal University, Wuhan, China, the M.S. degree in operational research from the Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree in computer science from Princeton University, Princeton, NJ.

He is a Senior Researcher with Microsoft Research, Redmond, WA. He has worked on a variety of topics, including combinatorial optimization, linked figure animation, and microphone array signal processing. His current research interests include activity recognition, face modeling and animation, and multimedia collaboration. Before joining Microsoft Research, he was with Silicon Graphics as a Member of Technical Staff for two years, where he developed a trimmed NURBS tessellator which was shipped in both OpenGL and OpenGL-Optimizer products. He has authored or coauthored over 70 papers in peer-reviewed international journals and conferences and holds over 40 granted patents. He has served in the technical committees for many international conferences. He was the co-chair of the 2003 ICCV Workshop on Multimedia Technologies in E-Learning and Collaboration, the technical co-chair of 2006 IEEE International Workshop on Multimedia Signal Processing, and the technical co-chair of 2010 International Conference on Multimedia and Expo. He is an associate editor of *Machine Vision and Applications*.