Action Search by Example using Randomized Visual Vocabularies

Gang Yu, *Student Member, IEEE,* Junsong Yuan, *Member, IEEE,* and Zicheng Liu, *Senior Member, IEEE*

***Abstract*—Because actions can be small video objects, it is a challenging problem to search for similar actions in the crowded and dynamic scenes when a single query example is provided. We propose a fast action search method that can efficiently locate similar actions spatio-temporally. Both the query action and the video datasets are characterized by spatio-temporal interest points. Instead of using a unified visual vocabulary to index all interest points in the database, we propose the randomized visual vocabularies to enable fast and robust interest point matching. To accelerate action localization, we develop a coarse-to-fine video subvolume search scheme, which is several orders of magnitude faster than the existing spatio-temporal branch and bound search. Our experiments on cross-dataset action search show promising results when compared with the state of the arts. Additional experiments on a 5-hour versatile video dataset validate the efficiency of our method, where an action search can be finished in just 37.6s on a regular desktop machine.**

***Index Terms*—Video Pattern Search, Action Search, Randomized Visual Vocabularies, Fast Branch and Bound Search, Random Indexing Trees.**

I. Introduction

Although text-based search techniques (for example, Google and Bing search) are powerful, they can only be applied to searching documents and text-annotated records with appreciable success and satisfaction achieved. For analyzing and searching video data, however, such a keyword-based video search is far from satisfactory, and oftentimes, could yield not-so-useful results. For example, since high-level semantic meaning is fairly difficult to be described by a limited set of keywords, keyword based video search is not suitable for detecting a specific human action or video event, e.g., detecting a pick-up action or a vehicle committing a hit-and-run crime.

This work addresses the human action search problem in natural videos: given an action example as a query, e.g., hand waving or picking up, the goal is to detect and accurately locate similar actions in the video dataset. Although a lot of previous works have been done on action recognition [51] [2] [5] [9] [12] [25] and action detection [14] [10] [7] [42], it still remains a difficult problem to query a large-scale video database using a single action example.

First of all, for action search, usually only a single query example is provided. In such a case, the amount of training data is extremely limited and only available at the time of query, whereas in action classification [2] [21] [41] and detection [14] [10] [52], a lot of positive and negative training examples can be leveraged. Therefore it is much more difficult to identify and locate a specific action example in videos. Furthermore, possible action variations such as scale changes, style changes and partial occlusions only worsen the problem, let alone cluttered and dynamic backgrounds.

Second, a video search engine must have a fast response time because otherwise the user experience would suffer. Unlike video event recognition [34], where the goal is to classify or rank pre-segmented video shots, action search is more difficult as we need to not only recognize the target action, but also locate it accurately, i.e., identify the spatio-temporal extent of the action in the video space. The accurate localization is of great importance especially for the crowded scenes where there are multiple people or moving objects. However, because actions can be small video objects, it is time consuming to locate them in the large video space. In general, for a dataset consisting of tens of hours of videos, such an action search process is expected to finish in just a few seconds.

Finally, a retrieval process typically prefers to enable user interactions, which allows the user to clarify and update their preferences. Thus, a practical action search system must have the flexibility to refine the retrieval results by leveraging the labels resulting from subsequent user feedback. Although relevance feedback is popular in image search, there is much less work that supports interactive action search.

To address the above challenges, we develop an action search system that addresses two key challenges in content based video search: (1) video indexing and (2) action searching. Each video is characterized by a collection of spatio-temporal interest points (STIPs) [1]. To enable fast matching of STIPs between the query action and the video dataset, effective Indexing is required. Bag-of-words (BoW) is a popular solution to index these interest points by clustering them hierarchically [21]. However, as there is only a single vocabulary, it provides one fixed way to quantize the feature space, and inevitably introduces quantization errors. Regardless of how the quantization is done, it typically results in loss of information. Instead of trying to find the best vocabulary as in some previous work [3] [21], we propose to use an ensemble of vocabularies to index the database. Multiple vocabularies provide multiple representations to the data. Thus, the variance of estimation can be reduced as we increase the number of vocabularies.

The second component of our search system is the action search part, i.e., spatially and temporally localize the retrieved

This work is supported in part by the Nanyang Assistant Professorship SUG M58040015 to Junsong Yuan.

Gang Yu and Junsong Yuan are with the School of EEE, Nanyang Technological University, 639798 Singapore

Zicheng Liu is with Microsoft Research Redmond, WA 98052-6399, USA

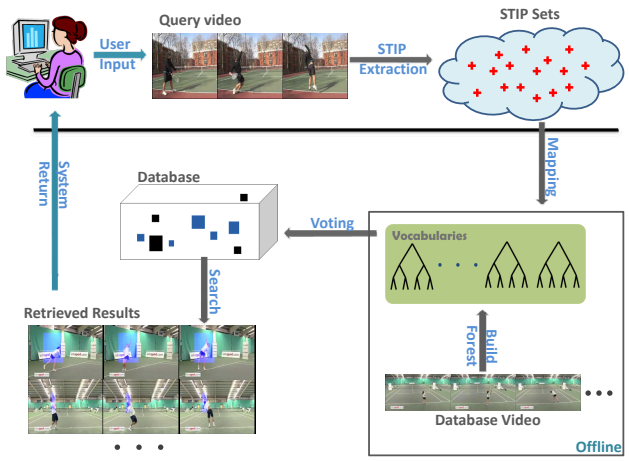

Fig. 1. Overview of our algorithm.

actions from the database. Our action search method is based on our previous work of spatio-temporal branch-and-bound search [7] [15] but we further significantly improve the localization speed by introducing a coarse-to-fine search scheme, which is several orders faster than [7].

An overview of our action search system is depicted in Fig. 1. The spatial-temporal interest points in videos are first extracted and then labeled according to the exemplar action in the query phase by our random indexing trees. The database can be considered as one large volume, with different values at the positions of the STIP points. The spatio-temporal video subvolumes of highest matching scores, i.e., the summation of scores from all of its interest points, are cropped out as detections. The subvolume is the retrieved result by our algorithm. We summarize the three contributions of our work below.

- We propose to index video interest point features using randomized visual vocabularies to compensate for information loss when using a single visual vocabulary. To implement the randomized visual vocabularies, we use random indexing trees, which provide fast indexing and leads to superior search results.
- For action localization, our proposed coarse-to-fine subvolume search strategy significantly improves the efficiency of the state-of-the-art action detection methods [7] [15], with comparable detection accuracy. With a single desktop computer, our method can search 5-hour long video within only 37.6 seconds.
- Our method does not rely on human detection, tracking, and background subtraction, and can handle action variations due to small scale and speed variations. It also supports interactive search by incrementally adding user labeled actions to the query set.

Experiments on cross-dataset search validate the effectiveness and efficiency of our proposed method.

II. Related Work

Action analysis, e.g., action recognition and action search, has a lot of practical uses in our daily lives. A lot of great works [34][30][11][37][43][35][33] have been done on this topic. Action classification attempts to determine the action category of the given video clips. In [10], a 3D Haar feature based optical flow is proposed to represent 3D volumes. [9] presents a maximum average correlation height filter, with which the intra-class variability is well captured. A visual spacetime oriented energy structure representation is proposed in [16], which is robust to scene clutter and rapid dynamics. In addition to these global template based action representations, local feature description [1] based representations have also been widely used. [2] employs the bag-of-words (BoW) models based on local feature points [1] and SVM as the classifier. Similar bag of words representations are also discussed in [3] [21]. Apart from BOW representations, Gaussian Mixture Models (GMM) [14] and nearest neighbor (NN) search [7] are alternative solutions to action classification. Different to sequence based action classification, [53][54] proposed to use "snippets", which is a very short sub-sequence (usually 1-7 frames). In [53], dense form and motion features are employed to describe the snippets and remarkable results have been achieved for action classification. [54] proposed an incremental learning framework based on "snippets" and well handled the challenging UCF sports dataset [9]. Our work differs from the "snippets" [53][54] in three aspects. First, we are doing different applications, action search (retrieval) versus action classification. Second, at the training stage, [53][54] tried to train a classifier which utilized all the label information from the training data while our work is doing an index job without any label information. Third, our work can spatial-temporally locate the actions without any other auxiliary tools. Other interesting action classification works include [25][32][36][39].

Different from action classification, action detection requires to locate when and where the desired action happened in the given video clips. This is more challenging due to the extremely large search space. One solution to this problem is sliding window. [10] presented a 3D spatiotemporal volumetric features and a cascade filters for efficient event detection. In [38], a video-to-video volume similarity with canonical correlation analysis is proposed and speeded up with the help of dynamic learning of the subspaces. [50] employed 3-dimension space-time volumes as similarity measure to locate the similar behaviors. The problem with sliding window is its high computational cost which can be relieved by branch and bound search [26]. Its extension version is introduced in [7]. In [7], videos are represented with a set of STIP points and, by LSH matching with the training dataset, each STIP point is weighted with a mutual information based model. Then branch and bound search is performed to search over the weighted video space. [14] proposed a MAP estimation framework which combines model adaptation and action detection. Because it explores the spatial-temporal coherence of actions and utilizes the prior information, it is effective to handle the problem of cross-dataset action detection. Several speeding up tricks for branch and bound search have been proposed in [15], including spatial-downsampling, Top-K search and λ search. In this paper, our system is built based on the branch and bound search but with even faster strategy compared

with [15]. Besides from sliding all possible windows, Hough voting is an efficient alternative to locating the actions, as shown in [29] [20]. Since the actions we want to detect are usually periodic which makes it hard to determine the temporal centers, Hough voting is currently difficult to well solve this problem.

Despite great successes in action recognition and detection, action retrieval, on the other hand, is less exploited. We can roughly categorize most of the existing action retrieval algorithms into two classes based on the number of query samples. Algorithms in the first category [16] [28] perform the sliding window search on the database with a single query sample. The idea for both [16] and [28] is to represent query and database videos with some features and to compare the similarity based on query-to-subvolume measurements. In [16], visual space-time oriented energy measurements are used while a five-layer hierarchical space-time model is employed in [28]. One limitation of these techniques is that with a single query sample, it is challenging to model action variations. Besides, an action retrieval system usually involves user interactions but their approaches do not have the capability to incrementally refine their models based on the user feedback. The other category of action retrieval algorithms, for example [19], is based on a set of query samples, usually including both positive and negative samples. Despite the fact that they work well in uncontrolled videos, the computational cost is high and they would fail if insufficient number of query samples are provided. Apart from the above work, there exist some other algorithms in the literature. For example, [13] [22] [23] rely on auxiliary tools like storyboard sketches, semantic words and movie transcripts for action retrieval, while [11] is specifically focused on quasi-periodic events. [24] performs action retrieval based on static images. [48] retrieves the similar human action patterns with spatiotemporal vocabulary.

III. Video Representation and Randomized Visual Vocabularies

We characterize a video by a set of spatial-temporal interest points (STIP) [1], denoted as $V = \{d_i \in \mathbb{R}^n\}$. Following [1], each STIP point d is described by two kinds of features: HOG (Histogram of Gradient) and HOF (Histogram of Flow) and the feature dimension n is 162. For action retrieval, we are given a database with N video clips. Denote the video clips as $\mathcal{V}_i, i = 1, \cdots, N$. Denote $\mathcal{D} = \{\mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_N\}$. These video clips contain various types of actions such as handwaving, boxing, and walking and last for several hours long.

In order to search for human actions in a large database, indexing becomes one of the most crucial parts. Traditionally, bag-of-words models [3] [21] with hierarchical K-means is widely used for interest points indexing. However, there is usually only a single vocabulary for BoW. The vocabulary quantizes the data in one fixed way. Quantization error would be introduced regardless of how we quantize the data. Intuitively, one vocabulary can be considered as one way of "viewing" the data. Instead of trying to find the best vocabulary as in the pervious work, we propose to use an ensemble of vocabularies. Multiple vocabularies can provide different viewpoints to the data. This can help to increase our performance as we increase the number of vocabularies.

To implement the vocabularies in an efficient manner, we try to employ the tree structures. Although there have been a lot of works on the tree structures for computer vision applications, little work has been done for efficient index with tree structures. KD-tree allows exact NN search but it is inefficient in high dimension cases and only slightly better than the linear search. Hierarchical K-means usually leads to unbalance trees and training is a time consuming process. To overcome the above problems, we propose the random indexing trees, which can explore the data distribution in the high dimension cases and index the database in an efficient and effective way.

Assume we have N_D STIP points in the dataset, denoted as $\{x_i = (x_i^1, x_i^2), i = 1, 2, \cdots, N_D\}$; $x_i^1 \in \mathbb{R}^{72}$ and $x_i^2 \in \mathbb{R}^{90}$ are the HOG feature and HOF feature, respectively. In order to build a tree and split the dataset, 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 HOG feature and $x_i^{\tau=2}$ means HOF feature.) Then two more random numbers e_1 and e_2 will be generated which are the dimension indices of the feature descriptor (either HOG feature or HOF feature depending on the value of τ.) After that, a "feature difference" can be evaluated with $D_i = x_i^\tau(e_1) - x_i^\tau(e_2), i = 1, 2, \cdots, N_D$. Based on all the D_i, we can estimate the mean and variance of the feature difference.

To put it briefly, a hypothesis (with variables τ, e_1 and e_2) can be generated with the following three steps:

- Generate $\tau \in \{1, 2\}$ to indicate the type of feature to use
- Generate the dimension indexes e_1 and e_2 and compute the feature difference $D_i = x_i^\tau(e_1) - x_i^\tau(e_2), i = 1, 2, \cdots, N_D$
- Split the dataset into two parts based on the mean of feature differences and obtain a variance

We generate γ hypotheses and find the one with the largest variance on feature difference. Since the performance is not sensitive to the number of hypotheses, we fix $\gamma = 50$ for all our experiments. Usually, a larger variance means that the data distribution spreads out more and the feature difference is more significant. Therefore the corresponding mean is used as the threshold to split the dataset. After this, one node will be built and the dataset 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 the predefined maximum depth is reached.

Compared with the local sensitive hashing (LSH) based indexing used in previous work [7], the benefits of random indexing trees are numerous. In this paper, we point out four properties that are essential for us. First, each tree in the model is almost independent from others. Second, our random indexing trees are fast to evaluate during the query stage. The computation time only depends on the number of trees and the depth of each tree. Hence, it is usually faster than LSH based nearest neighbor search [7]. In the experiments, we will show that our random-indexing-trees based weighting approach is over 300 times faster than LSH based approaches.

This is of great importance if we want to perform real-time action analysis. Another advantage of random indexing trees compared with LSH is that, during the construction of each tree, data distribution of the STIPs is integrated, which means the tree construction is guided by the data density. This is one reason why random indexing trees has great speed gain but little performance loss. Finally, by adding more trees, we can alleviate the effect of lacking query samples and well model the intra-class variations. As shown in Fig. 2, for each vocabulary, only a small portion of nearest neighbors can be found. The benefit of multiple visual vocabularies is to quickly find enough nearest neighbors to boost the confidences of multiple matches.

IV. Action Matching using Randomized Visual Vocabularies

We can consider action search as a template matching process. That is, matching the spatio-temporal template (query STIP points) with all the video sub-volumes in the database. More specifically, our objective is, given one or more query videos, referred to as $\mathcal{Q}$, to extract all the sub-volumes which are similar to the query. Formally, that is to find:

$$V^* = \max_{V \subset \mathcal{D}} s(\mathcal{Q}, V), \tag{1}$$

where $s(\mathcal{Q}, V)$ is a similarity function between a set of query video clips $\mathcal{Q}$ and a subvolume V in the database.

Unlike previous single template action detection and retrieval [16], which can only take one positive sample for query, our approach can integrate multiple query samples and even negative ones. By introducing negative samples during the query phase, our algorithm is more discriminative. In addition, this approach enables interactive search by leveraging the labels obtained from user feedbacks.

Following our previous work in [7], we use the mutual information as the similarity function for $s(\mathcal{Q}, V)$. So we have:

$$\begin{aligned} V^* &= \max_{V \subset \mathcal{D}} MI(\mathbf{C} = c_{\mathcal{Q}}, V) \\ &= \max_{V \subset \mathcal{D}} \log \frac{P(V|\mathbf{C}=c_{\mathcal{Q}})}{P(V)} \\ &= \max_{V \subset \mathcal{D}} \log \frac{\prod_{d_i \in V} P(d_i|\mathbf{C}=c_{\mathcal{Q}})}{\prod_{d_i \in V} P(d_i)} \\ &= \max_{V \subset \mathcal{D}} \sum_{d_i \in V} \log \frac{P(d_i|\mathbf{C}=c_{\mathcal{Q}})}{P(d_i)}, \end{aligned} \tag{2}$$

where $\mathbf{C} = c_{\mathcal{Q}}$ refers that we want to find the sub-volumes similar to the query clip $c_{\mathcal{Q}}$.

We refer to $s^{c_{\mathcal{Q}}}(d_i) = \log \frac{P(d_i|\mathbf{C}=c_{\mathcal{Q}})}{P(d_i)}$ as the mutual information between STIP d_i and query set $\mathcal{Q}$. In [7], $s^{c_{\mathcal{Q}}}(d_i)$ is computed based on one positive nearest neighbor point and one negative nearest neighbor point from d_i. However, nearest neighbor search in high dimensional space is very time consuming even with the advanced local sensitive hashing (LSH) technique [7]. Second, this approach is sensitive to noise, since only two points are used to compute its score. In order to address these problems, we formulate $s^{c_{\mathcal{Q}}}(d_i)$ as:

$$\begin{aligned} s^{c_{\mathcal{Q}}}(d_i) &= \log \frac{P(d_i|\mathbf{C}=c_{\mathcal{Q}})}{P(d_i)} \\ &= \log \frac{P(d_i|\mathbf{C}=c_{\mathcal{Q}})P(\mathbf{C}=c_{\mathcal{Q}})}{P(d_i)P(\mathbf{C}=c_{\mathcal{Q}})} \\ &= \log \frac{P(\mathbf{C}=c_{\mathcal{Q}}|d_i)}{P(\mathbf{C}=c_{\mathcal{Q}})}. \end{aligned} \tag{3}$$

In Eq. 3, $P(\mathbf{C} = c_{\mathcal{Q}})$ is the prior probability that can be computed as the ratio of the number of positive query STIPs to the total number of query STIPs.

In order to estimate $P(\mathbf{C} = c_{\mathcal{Q}}|d_i)$ efficiently and robustly, random indexing trees are used. We consider each tree as one partition of the data space. In the following section, we discuss how to estimate $P(\mathbf{C} = c_{\mathcal{Q}}|d_i)$ given multiple random indexing trees.

Suppose N_T random indexing trees have been built offline from the database. At the query stage, all the STIP points in the query set $\mathcal{Q} = \mathcal{Q}_P \cup \mathcal{Q}_N$ (where $\mathcal{Q}_P$ and $\mathcal{Q}_N$ refer to positive query and negative query, respectively) are first extracted and distributed into the trees. Fig. 2 gives a two-dimension example where blue and black dot points represent the positive and negative STIPs, respectively. Each STIP point $d_i \in \mathcal{D}$ (red square in Fig. 2) falls into one of the leaves of a tree. Each leaf node contains several STIP points $d_q \in \mathcal{Q}$. In order to compute the posterior $P(\mathbf{C} = c_{\mathcal{Q}}|d_i)$, we integrate the information from all the leaves which contain d_i. Suppose d_i falls into a leaf with N_k^+ positive query STIP points and N_k^- negative points for tree T_k, then $P(\mathbf{C} = c_{\mathcal{Q}}|d_i)$ can be computed as:

$$P(\mathbf{C} = c_{\mathcal{Q}}|d_i) = \frac{1}{N_T} \sum_{k=1}^{N_T} \frac{N_k^+}{N_k^+ + N_k^-}. \tag{4}$$

As can be seen from Eq. 4, our voting strategy can integrate negative query samples, which makes our algorithm more discriminative.

Eq. 3 can hence be rewritten as:

$$\begin{aligned} s^{c_{\mathcal{Q}}}(d_i) &= \log P(\mathbf{C} = c_{\mathcal{Q}}|d_i) - \log P(\mathbf{C} = c_{\mathcal{Q}}) \\ &= \log \tfrac{1}{N_T} \sum_{k=1}^{N_T} \tfrac{N_k^+}{N_k^+ + N_k^-} - \log P(\mathbf{C} = c_{\mathcal{Q}}). \end{aligned} \tag{5}$$

However, in the case where there are no negative query samples available ($\mathcal{Q}_N = \emptyset$), we slightly modify Eq. 4 to:

$$P(\mathbf{C} = c_{\mathcal{Q}}|d_i) = \frac{1}{N_T} \sum_{k=1}^{N_T} \frac{N_k^+}{M}, \tag{6}$$

where M is a normalization parameter. And Eq. 5 can be written as

$$\begin{aligned} s^{c_{\mathcal{Q}}}(d_i) &= \log \tfrac{1}{N_T} \sum_{k=1}^{N_T} \tfrac{N_k^+}{M} - \log P(\mathbf{C} = c_{\mathcal{Q}}) \\ &= \log \tfrac{1}{N_T} \sum_{k=1}^{N_T} N_k^+ - \log M - \log P(\mathbf{C} = c_{\mathcal{Q}}). \end{aligned} \tag{7}$$

We denote $A = -\log M - \log P(\mathbf{C} = c_{\mathcal{Q}})$. A is a parameter which is set empirically.

Each tree is a partition of the feature space as shown in Fig. 2. Hopefully, STIP points in the same leaf node are similar to each other. The score evaluation (Eq. 5) on the trees can be explained intuitively by a dyeing process. We can think of

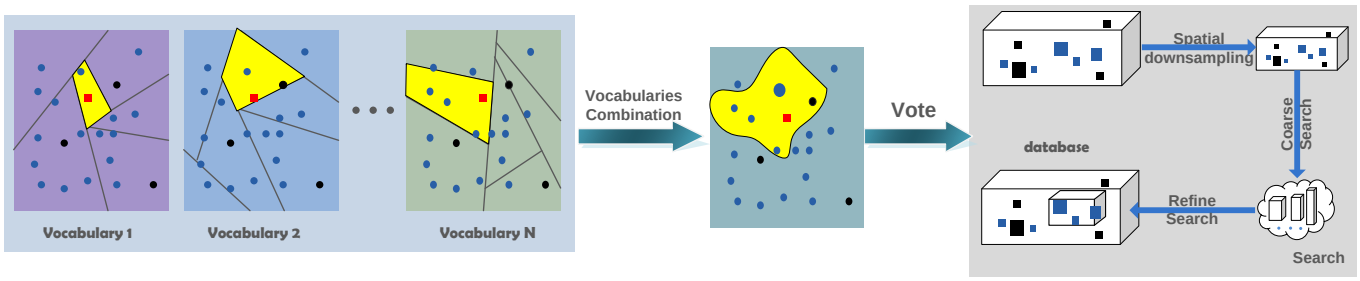

Fig. 2. A schematic illustration of randomized-visual-vocabulary indexing and action search.

each positive query STIP point as having a blue color and a negative point as having a black color. For each query point, we pass it down each tree. The leaf that the point falls in is dyed in the same color as the query point. Each leaf keeps a count of the number of times it is dyed by blue and a count of the number of times it is dyed by black after we pass all the positive and negative query points down the trees. If a leaf's blue count is larger than the black count, it is more likely to belong to the positive region, and vice versa. Similarly, if the blue count is the same as the black count, it is more likely not to vote any side, i.e., vote zero. Given a point d_i (red square point in Fig. 2) in the dataset, to compute its score with respect to the positive queries, we pass it down each tree. From each tree, we find the leaf that d_i falls in. The blue counts and black counts of all the leafs in all the trees that d_i falls in are combined to estimate its posterior $P(\mathbf{C} = c_{\mathcal{Q}}|d_i)$, as given in Eq. 5. In some sense, the random indexing trees are like a special kernel, as shown by the yellow regions in Fig. 2. The idea of dyeing process is not only limited to trees but also applicable to any vocabulary structure.

Even though our random indexing trees share some similar properties with random forest, e.g. [4][20], there are important differences between our technique and random forests. First, our random indexing trees are constructed in an unsupervised manner for class-independent video database indexing, while traditional random forests are constructed in a supervised manner. Second, our random indexing trees generate both positive and negative voting scores, thus it is more discriminative compared to [4][20], which generates only positive votes based on the frequency. Third, we use random indexing trees for density estimation, which has not been exploited before.

V. Efficient Action Search

A. Coarse-to-fine Hierarchical Subvolume Search Scheme

After computing the scores for all the STIP points in the database, we follow the 3D branch and bound approach in [7] to search for subvolumes in each video in the database. The idea of branch and bound search is to branch the search space and give an upper bound to each candidate subset. The candidate subsets can be dropped if the upper bound is lower than the current optimal value. In [7], 3D branch and bound search was proposed and to decompose the search space into the spatial temporal domain, respectively. However, as stated in [15], there are two limitations in the subvolume search method proposed by [7]. First, we need to run multiple rounds of branch and bound search if we want to detect more than one instance. In addition, the computational cost is extremely high when the video resolution is high.

In [15], three speeding up techniques have been proposed to reduce the computational cost of branch-and-bound search. They are spatial-downsampling, λ search and Top-K search. Although significant computational advantages have been achieved, it stills takes 26 minutes to search one hour video database. This is unacceptable for action retrieval application.

To further reduce the computational cost, a coarse-to-fine hierarchical search is proposed here. The basic idea is to first search the coarse resolution score volumes to quickly find a number of candidate subvolumes, and then refine the candidate subvolumes using the finer resolution score volumes. Note that this technique is different from [15] in that we introduce a refinement mechanism so that the results obtained at a coarser resolution are refined in a higher resolution score volume.

According to [15], the computational complexity of the top-K search is $O(m^2n^2t) + O(Kmnt)$, where m, n, t are the width, height and duration of the database video and K refers to the top K results. Obviously, the most effective way to reduce the computational cost is to reduce the spatial resolution of the score volume. Thus, spatial-downsampling is performed to compress the search space. The following error bound for score volume downsampling is proposed in [15].

$$f^s(\tilde{V}^*) \geq (1 - \frac{s*h + s*w + s^2}{wh}) f(V^*), \tag{8}$$

where $\tilde{V}^* = argmax_{V \in \mathcal{D}^s} f^s(V)$ denotes the optimal subvolume in the downsampled search space $\mathcal{D}^s$, $f(V) = MI(\mathcal{Q}, V)$, and V^* refers to the optimal subvolume in the original search space with width w and height h. With the help of this error bound, we can relax the top searched list to include more results for a further round of re-ranking. The benefit of using this error bound is to reduce the precision loss in the coarse round of search. Suppose we want to eventually retrieve the top K results from the database, we first downsample the score volumes with a factor of $2a$ and retrieve the top $\hat{K}$ subvolumes by employing the top-K search algorithm [15]. We choose $\hat{K} = 2K$ in our experiments.

After that, we can estimate a threshold, denoted as θ, based on the Kth largest subvolume score (denoted as $f^s(\tilde{V}^{(K)})$.) For example, if we set downsampling factor $s = 8$ and assume $w = h = 64$, then our approximation has an average error:

$$\frac{s*h + s*w + s^2}{wh} = 56.3\%. \tag{9}$$

So we choose $\theta = 0.437 f^s(\tilde{V}^{(K)})$ to filter the first round

results. Then, for each remaining subvolume $\tilde{V}^{(k)}$ from the first round, we first extend the spatial size with 30 pixels in each direction and another round of branch-and-bound search is performed. Different from previous round, which is running over the entire video space, this round of search is performed over the filtered 3D-subvolumes (extended with 30 pixels in each spatial direction). λ search [15] with $\lambda = f^s(\tilde{V}^{(K)})$ and downsampling factor a is used in this round of search.

Despite that only two rounds of search are used, our algorithm can be extended to more rounds of search. This is especially useful to handle high-resolution videos. As shown in Table V, our efficient two-round branch-and-bound search only costs 24.1 seconds to search a database of one hour long 320×240 videos. This is over 60 times faster than the approach in [15] and 2900 times faster than [7]. Even for a 5-hour large database, it only costs 37.6 seconds to respond to the users. Besides the speed advantages, we will see from the experiments that the search accuracy is not compromised.

Finally, we differentiate our work from [15] here. In [15], only one round of search is employed. The search is performed in the entire three-dimensional video space. However, in our work, with the help of the coarse round search, our fine round search only need to focus on the potential regions, which can save significant computational cost. Besides, to reduce the computational loss from the coarse round search, the error bound is used to determine the number of candidate sub-volumes for re-ranking.

B. Refinement with Hough Voting

Although our search algorithm can successfully locate the retrieved actions, the localization step may not be accurate enough, as can be seen from the first row of Fig. 4. This motivates us to add a refinement step. The idea is to back-project the initial sub-volume into the query video. Based on the matches of STIPs, we can vote the action center (only in the spatial domain). Fig. 3 is an illustration of the Hough refinement step.

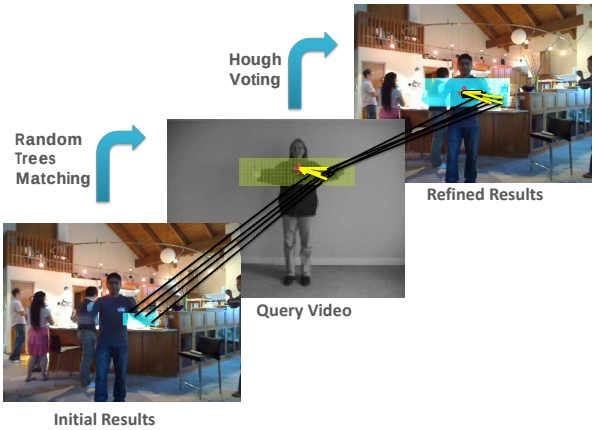

Fig. 3. Illustration for Hough refinement.

Suppose we already have the initial results from the down-sampled branch and bound search (the blue region in the left image of Fig. 3), for all the STIP points within the detected subvolume, we match them with the STIPs in the query video clip, either by trees or Nearest Neighbor search. Then the shift from the matched STIPs in the query will vote for the center of the retrieved action. To simplify the problem, we only consider one fixed scale and smooth the votes with a Gaussian kernel. After considering all the votes, the center of the retrieved action is the position with the largest vote (the red cross in the third image of Fig. 3). To recover the spatial extent, we set the spatial scale of the action to be the smallest sub-volume which includes the initial retrieved region and the temporal scale is fixed to the initial retrieved result.

After the refinement, the blue region in the right image of Fig. 3 gives an illustration of the revised result compared with original round of result, i.e. the left image. Both quantitative results, as shown in Fig. 7, and empirical results in Fig. 4 show that the refinement step can successfully improve our retrieved results. The major steps of our algorithm are described at Algorithm 1.

C. Interactive Search

The performance of our action retrieval system is constrained by the limited number of queries. To show that our retrieval system can achieve better results when more queries are provided, we add an interaction step to facilitate human interaction. There are two major advantages of the interaction step. The first is to allow the user to express what kind of action he/she wants to retrieve. Another advantage is that our system can benefit from more query samples after each round of interaction.

To implement the system, we first perform one round of action retrieval based on a few query samples. After that, the user would label D (D=3 in our experiments) detections with the highest scores. Then the D newly labeled subvolumes will be added into the query set for the next round of retrieval. Detailed results will be discussed in the experiment section.

Algorithm 1 The proposed algorithm for action search

Input:

- Database with the random indexing trees: $\{T_k\}$
- Query video: $\mathcal{Q}$

Output:

The top K retrieved sub-volumes $\{V_1, V_2, \cdots, V_K\}$

1: **Voting:** Based on Eq. 7, vote the database STIP points by the random indexing trees (Section IV).
2: **Coarse-round search:** Retrieve the top $\hat{K}$ subvolumes by employing the top-K search algorithm for each video in the database. The downsampling factor is set to $2a$. Then keep those detected sub-volumes with scores above the threshold θ (discussed after Eq. 9) only.
3: **Fine-round search:** Rerank the remaining sub-volumes from the first round of results with branch and bound search (downsampling factor a).
4: **Refinement:** [optional] Hough refinement discussed in Section V-B.

Fig. 4. Comparison of retrieval results without and with Hough refinement. For each row, the first image indicates the query sample and the following 7 images refer to the highest ranked retrieved results. All the experiments are done with only one query sample without user feedback. The upper and lower rows are the experiments on handclapping without and with Hough refinement, respectively. The query clip (first column) is from KTH while the database is from MSR II.

D. Computational Complexity

For our action retrieval system, there are two major runtime costs: voting and searching. The computation complexity is $O(N_s T_d N_T)$ for the voting step where N_s refers to the number STIPs in a query clip, T_d refers to tree depth, and N_T refers to the number trees in a forest. As shown in the Table V, the voting time is negligible compared to the search time. For action search, the worst time complexity is

$$\begin{aligned} T = \quad & O((m/2a)^2(n/2a)^2 t) + O(\hat{K}(m/2a)(n/2a)t) \\ & + O((\hat{m}/a)^2(\hat{n}/a)^2\hat{t}) + O(K(\hat{m}/a)(\hat{n}/a)\hat{t}) \end{aligned} \tag{10}$$

where m, n and t are width, height and duration of the clips in database. a is the downsampling factor and $\hat{K}$ (this value depends on the retrieval scores) is a little larger than K ($K = 7$ refers to the number of retrieved results in our experiment). After a filtering step (the first two complexity), $\hat{m}, \hat{n}, \hat{t}$ are used to represent the spatial width, spatial height and temporal duration for remaining sequences (usually $\hat{t} << t$), respectively. The quantitative analysis of the computational cost will be discussed in the experimental part.

VI. Experimental results

To validate our proposed algorithm, seven experiments on six datasets have been discussed in this section. KTH [3] and MSR II [7] are used to evaluate our algorithm for action classification and action detection, respectively. Table III lists the five datasets for the validation of our search algorithms. Sample frames from the five datasets can be found in Fig. 5. Since our algorithm is trying to handle the action search problem, we focus on the action search experiments in this section. In order to provide a quantitative comparison with other work, we give an action retrieval experiment on benchmark dataset MSRII first. After that, illustrative experiments for action search on CMU dataset [6], Youtube videos, and UCF dataset [9] are discussed. To show the ability to handle real action retrieval by our system, we build a 5-hour dataset with videos downloaded from datasets MSRII [7], CMU [6], VIRAT [46], Hollywood [47] and some videos downloaded from Youtube.

A. Action Classification on KTH

We first give an experiment to show that our algorithm is able to handle the traditional action classification problem. We use the benchmarked KTH dataset [3] and test our algorithm with the same setting as in [3][15]: 8+8 sequences for training/validation and 9 sequences for testing. Table I lists the confusion matrix for our algorithm and a comparison with other works are listed in Tabel II. Although our work aims to handle the action search problem, i.e., the label information from the training data is not utilized until the testing stage, we still achieves comparable results.

	walk	clap	wave	box	run	jog
walk	142	0	0	0	0	2
clap	0	136	1	7	0	0
wave	0	11	133	0	0	0
box	4	0	0	140	0	0
run	1	0	0	0	114	29
jog	2	0	0	0	26	116

TABLE I
CONFUSION MATRIX FOR KTH ACTION DATASET. THE TOTAL ACCURACY IS 90.4%.

Method	Average accuracy
Our method	90.4%
[15]	91.8%
[5]	90.3%
[3]	91.8%
[14]	95.02%
[54]	94.4%
[25]	94.53%

TABLE II
COMPARISON OF DIFFERENT REPORTED RESULTS ON KTH DATASET.

B. Action Detection on MSR II

We then validate our random-indexing-trees strategy with a challenging action detection experiment. Since handwaving is an action that is quite common in real life, we choose to detect handwaving actions in this experiment. We first train the model with KTH dataset (with 16 persons in the training part) and then perform experiments on a challenging dataset (MSR II) 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.

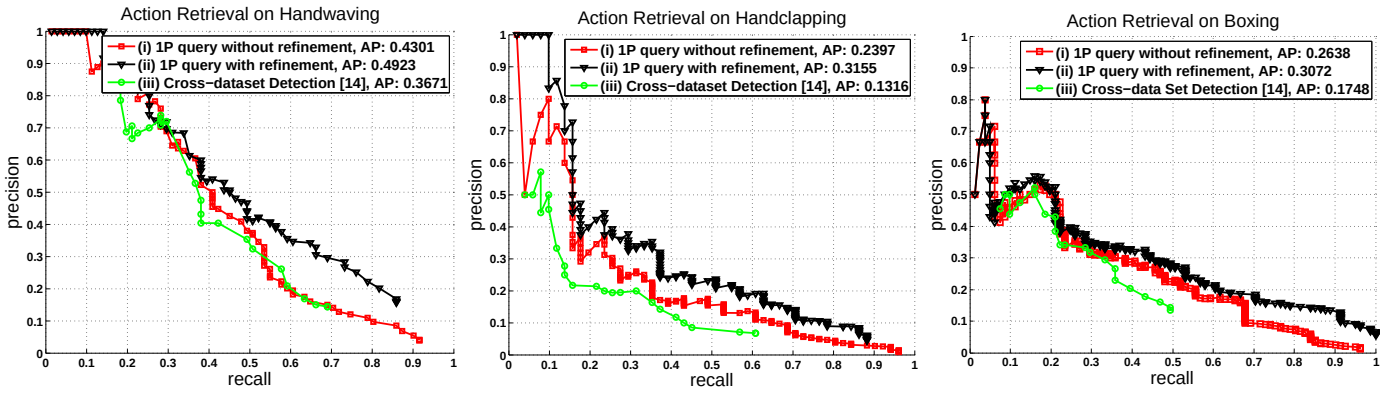

Fig. 7. Precision-recall curves for action search on MSR II.

Dataset	Total Length	Query Action
MSR II	1 hour	Waving, Clapping, Boxing
CMU	20 mins	Waving, Bending
Youtube	4.5 mins	Tennis Serve
UCF Sports	$\approx$ 15 mins	Diving, Weightlifting
Large Dataset	5 hours	Waving, Clapping, Boxing, Ballet Spin

TABLE III
LIST OF DATASETS FOR EXPERIMENTS

Fig. 5. Sample frames from our testing datasets. The first to the third rows show sample frames from MSR II, CMU and Youtube videos, respectively. The fourth row shows some different frames from the 5-hour large dataset.

Fig. 6 compares the precision-recall curves for the following methods (the resolution for the original videos is 320×240):

(i) ASTBB (Accelerated Spatio-Temporal Branch-and-Bound search) [8] in low resolution score volume (frame size 40×30),
(ii) Multi-round branch-and-bound search [7] in low-resolution score volume (frame size 40×30),
(iii) Top-K search in down-sampled score volume discussed in [15] (size 40×30, but for the indexing we do not use the supervised trees in [15]),
(iv) ASTBB [8] in 320×240 videos,
(v) Random-indexing-trees based voting followed by Top-K search in down-sampled score volume (size 40×30).

The first four methods ((i)-(iv)) employ the LSH based voting strategy [7]. The measurement of precision and recall is the same as those described in [7]. To compute the precision

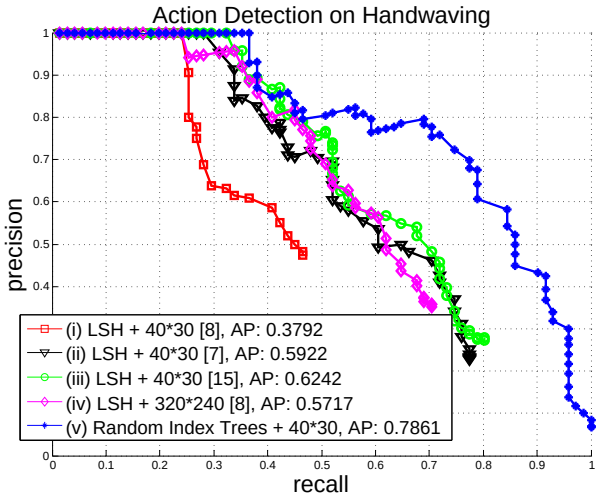

Fig. 6. Precision-recall curves for handwaving detection on MSR II. AP in the legend means the average precision.

we consider a true detection if : $\frac{\text{Volume}(V^* \cap G)}{\text{Volume}(G)} > \frac{1}{8}$, where G is the annotated ground truth subvolume, and V^* is the detected subvolume. On the other hand, to compute the recall we consider a hit if: $\frac{\text{Volume}(V^* \cap G)}{\text{Volume}(V^*)} > \frac{1}{8}$. According to Fig. 6, our random-indexing-trees based action detection outperforms the other algorithms. Compared with LSH voting strategy ((i)-(iv)), it shows that our random-indexing-trees based voting is more discriminative and robust. The underlying reason is that our random trees are data-aware, i.e. we model the data distribution when constructing the trees. Besides, since LSH only uses two nearest neighbors for voting, the results are easily corrupted by noise. In [15], random forest is employed to perform action detection. However, the difference is that trees in [15] are constructed in a supervised manner, which means that the label information is utilized when splitting the nodes, while our random trees are unsupervised built with the purpose of modeling the underlying data distribution.

C. Action Retrieval on MSR II

To give a quantitative result for our action retrieval system, we use videos from MSR II as the database. The query samples are drawn from KTH dataset. As there hasn't been

any reported action retrieval results on MSR II dataset, we compare our retrieval results with several previously reported action detection results on this dataset. The evaluation is the same as that for action detection. For the implementations of our random indexing trees, we set the number of trees in a forest $N_T = 550$ and the maximum tree depth to 18. Fig. 7 compares the following three strategies on handwaving, handclapping and boxing actions (for the boxing action, we flip each frame in the query video so that we can retrieve the boxing coming from both directions[1]), respectively.

(i) One positive query example without Hough refinement,
(ii) One positive query example with Hough refinement,
(iii) Cross-Dataset detection [14][2],

As shown in Fig. 7, with a single query, our results ((i) and (ii)) are already comparable to (iii) for all three action types. This is quite encouraging because (iii) used all the training data while we only use a single query. Besides, our Hough refinement scheme (ii) improves the results without Hough refinement (i).

Fig. 8 shows the experimental results of interactive action retrieval. The following six strategies (all of them are performed without Hough refinement) are compared.

(i) One query example with random indexing trees (RIT) based voting,
(ii) One query example with LSH based voting,
(iii) One positive and one negative query examples
(iv) Two positive and two negative query examples,
(v) One iteration of user interaction after (i),
(vi) Two iterations of user interaction after (i).

Performance for LSH based indexing scheme [7] is listed with the similar framework as random indexing trees. The parameters for LSH are set to make the comparison fair. We can see that when there is only one query example, our random-indexing-trees based voting strategy (i) is superior to LSH based voting strategy (ii). When there are two query examples (one positive and one negative,) the retrieval results become worse than the one query case. The reason is that negative action type is more difficult to describe and a single negative example is sometimes not enough. Fig. 9 shows that we can increase our average precision by increasing the number of negative queries. The results would be stable if we use around four negative queries along with one positive query. Besides, the performance of our system increases as more query samples are given by interaction. In particular, after two interaction steps, our retrieval results are better than the results obtained by other action detection systems ((i)-(iv) in Fig. 6), which utilize all the training data (256 examples).

We also provide some illustrative results in Fig. 4. For each query, seven subvolumes with the highest scores are listed in the figure. The retrieved subvolumes are marked by colored rectangles. The rectangle with cyan background indicates a "correct" retrieval. As shown in the first row of Fig. 4, some of the cyan results are focused on a subregion of the action region. But this can be relieved with Hough refinement as indicated in the second row. In short, our action retrieval system can get very good results among the top retrieved subvolumes on various actions types.

[1]Only for the boxing action, we use the query video as well as the flipped version. For other actions, only the query video is used.

[2]The STIP features in [14] are extracted in video resolution of 160×120 but 320×240 for other methods

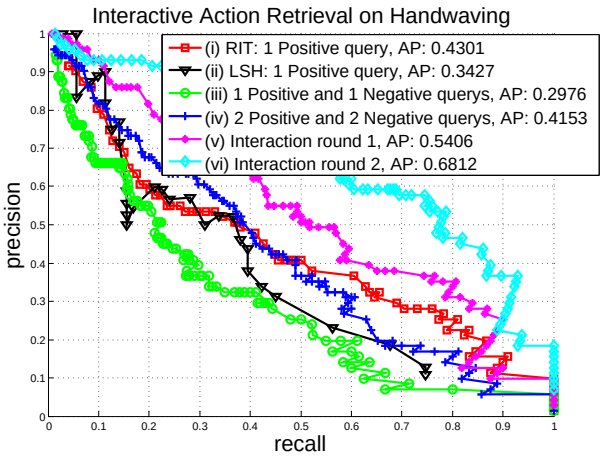

Fig. 8. Precision-recall curves for the interactive action retrieval on MSR II.

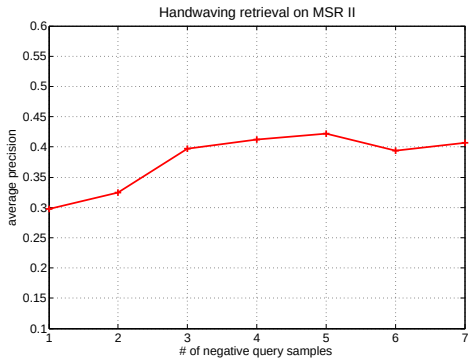

Fig. 9. Comparison of average precision based on the input of one positive query but different number of negative queries.

D. Action Retrieval on CMU Database

CMU database [6] is another widely used database for action analysis. Since the annotation of the actions includes the entire human rather than the action itself (as can be seen from Fig. 10, our results only mark the region where the action happens), we only give some illustrative examples on this dataset. The CMU database includes 48 videos of total duration around 20 minutes. The resolution for these videos are 160×120. Handwaving and bending actions are retrieved from the database where the query video for handwaving is from KTH and the query video for bending is from Weizmann dataset [45].

Fig. 10 shows the search results. For each row, the first image is a sample frame from the query video and the following 7 images are from the top-7 retrieved segments, respectively. The cyan region shows the positive detection while the yellow region shows the negative detection. Compared with handwaving, bending is a non-periodic action, which is more

challenging due to simple motion pattern and small number of query STIP points. From this experiment, we can see that our algorithm can handle the two actions with a large intra-class variations and clutter background, even in the low-resolution and highly compressed videos.

E. Action Retrieval on Youtube Video

In this experiment, we validate our algorithm with a challenging tennis serve action search from a Youtube video[3], which is also a non-periodic action. More action searches from Youtube videos will be available from our project website. The length for the database video is around 280s, with several tennis serving actions performed by different actors under different views. The query video is a 2 second segment cut from a different Youtube video[4]. The experiment is very challenging due to the following aspects. First, there are different scenes and players compared with the query clip. Besides, the serving actions are recorded in several different views. Second, it contains not only the serving action but also a lot of other actions as well. We need to differentiate the serving action from other actions. Third, in addition to the large intra-class variations, the visual quality is poor due to video compression. The regions marked blue (the reason to use blue color is that it differentiates with the background color) from 2rd to 6th rows of Fig. 11 are the top 5 retrieved sub-volumes based on the query video from the first row. We can see that our algorithm achieves promising results.

F. Action Retrieval on UCF Sports Database

We further validate our algorithm with UCF sports dataset [9]. We choose to search diving and weightlifting actions because there are no large camera motions. Since the videos have already been segmented, no localization is needed. Fig. 12 shows our experimental result. The two rows illustrate diving and weightlifting actions, respectively. For each row, the first column is a sample frame from the query video, and the subsequent five columns are the top-5 retrieved results. Although only a single query sample is used for search, we still obtain quite promising results.

G. Action Retrieval on Large-scale Database

To verify that our algorithms can handle large scale dataset, we build a large database with more than 200 videos. The database includes videos from datasets MSRII [7], CMU [6], VIRAT [46], Hollywood [47] and some videos downloaded from Youtube. The total duration is around 5 hours.

Four different actions (handwaving, handclapping, boxing and ballet spinning) are tested in this large dataset. Each experiment is done with only one query video, without any post-processing, e.g. Hough refinement. The query videos for the first three actions (around 15s for each action) are collected from KTH while the query video for ballet spinning is downloaded from Youtube (around 5s). For handwaving, handclapping, and boxing, we retrieve top-40 detections. For ballet spin, we retrieve top-10 detections since there are not as many ballet spin actions in the database. Fig. 13 shows five samples of the retrieved results of handwaving. The first row gives seven frames from the query video while the second to fifth rows show the four positive results where the retrieved subvolumes are marked with cyan color. The sixth row shows one negative result where the retrieved subvolumes are marked with yellow color. Similarly, Fig. 14, Fig. 15 and Fig. 16 show the results of handclapping, boxing and ballet spin, respectively. Besides, in Fig. 17, we give the retrieved performance (precision versus the number of top samples retrieved) for the large database. Based on the illustrative results, we can see that our algorithm can well handle the large scale changes, clutter background, partial occlusion and low visual quality.

H. Implementation Issues

To implement such a system, there are several issues we need to take care of in both the indexing stage and query stage. For indexing part, we need to determine the number of trees. We use an experiment to evaluate the relationship between the number of trees and average precision. The test environment is the same as that discussed in Section. VI-C. According to Fig. 18, the number of trees become stable from 300 for the handwaving and boxing action and from 500 for the handclapping action. Besides, if we use only one tree (traditional BOW model), the system cannot find any positive detections. The reason is that only a very limited number of database STIP points are matched to the query STIP points and these matched STIP points are isolated from each other. Usually only the STIP point itself forms a detection. Hence, we need to increase the number of trees to introduce more matches. But as shown in the figure, the performance will be stable when sufficient number of trees are used.

Based on Fig. 18, we fix the number of vocabularies (trees) as 550 in our experiment setting. The depth for the trees is set as 18 for most of our experiments (For Youtube video dataset, we set it as 15 because the database is of small size). In the query stage, the downsampling factor of the branch-and-bound search (referring as a) is first set to 16 in the coarse round of search and then refined as 8 for another round of search.

I. Computational Cost

In offline stage, we need to first extract the STIP features from the database videos. The time cost depends on the content of the videos, video resolution and video duration. The code for STIP feature is downloaded from the author"s website [1]. After that, it costs around 3 hours to build a vocabulary with 550 trees on the STIP features for MSR II dataset. We do not perform other pre-processing on the dataset. For the online cost, there are two major runtime costs: voting and searching. The total computational cost for our system is listed in Table V. The testing environment is as follows. We use one query video, which is approximately 20 seconds long. Two datasets are tested in our experiments. The first database is MSR II, which consists of 54 sequences with 320×240 resolution. The second dataset contains 5

[3]http://www.youtube.com/watch?v=inRRaudOf5g

[4]http://www.youtube.com/watch?v=NQcmYTIrqNI

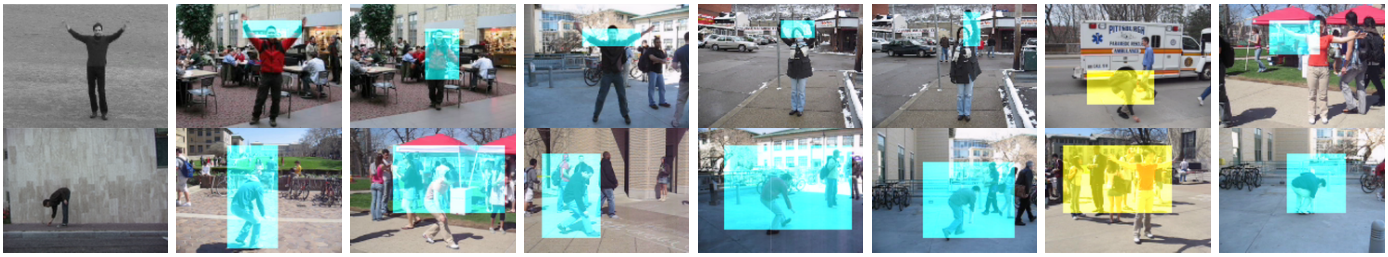

Fig. 10. Retrieval results for CMU dataset. For each row, the first image indicates the query sample and the following 7 images refer to the highest ranked retrieved results. All the experiments are done with only one query sample without any user feedback. For the first row, the query clip with handwaving action is from KTH while the database is from CMU [6]. For the second row, the query clip with bending action is from Weizmann Dataset while the database is from CMU [6].

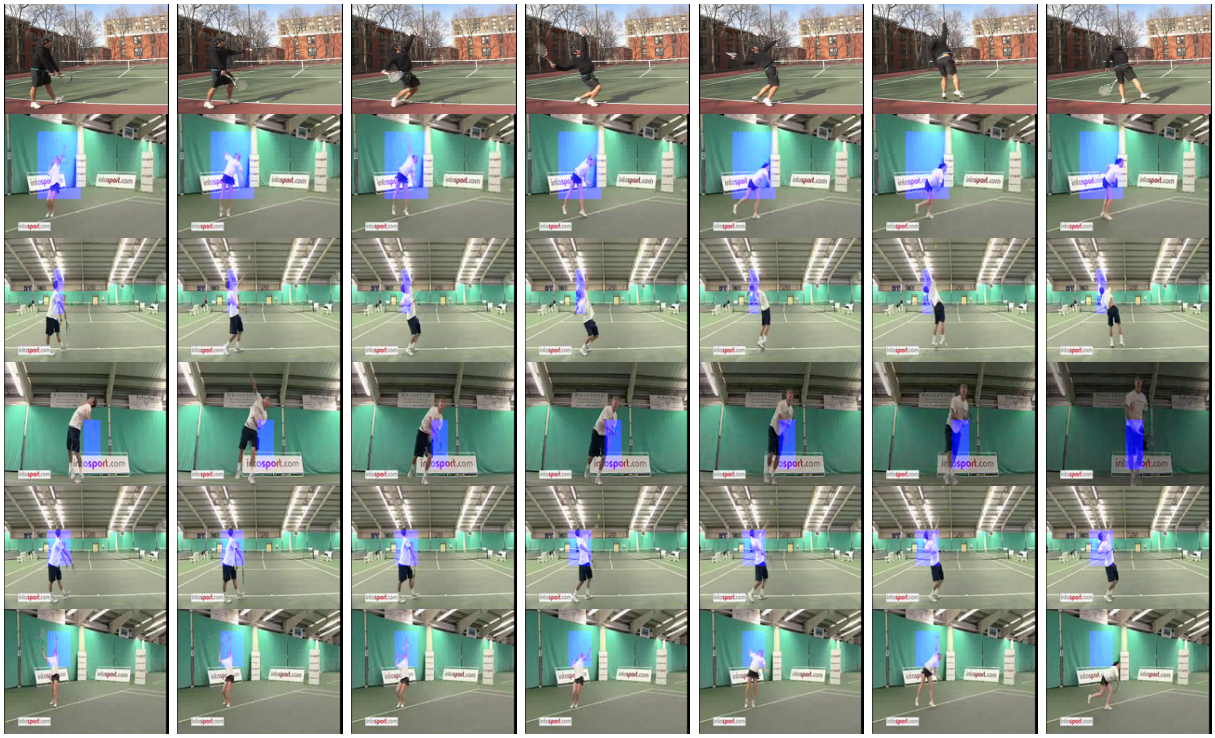

Fig. 11. An illustrative example to search tennis serve action. The query and database videos are downloaded from Youtube. The seven images in the first row show different frames in the query video while the images from 2rd to 6th rows show the top-5 searched results.

hours of videos (discussed in Section. VI-G). To set the parameter θ in Section V-A, we average the w and h among the top K results and obtain an error bound based on the estimated w and h. With this error bound, we can compute θ similarly as in Eq. 9. We use a PC with 3GHz CPU and 3G memory. Table IV compares the voting cost: the random-indexing-trees based vocabulary implementation is much more efficient compared with LSH. According to Table IV, random-indexing-trees method is over 300 times faster than LSH based indexing but with even superior performance from Fig. 8. For the searching cost, as shown in Table V, our coarse-to-fine subvolume search scheme only costs 24.1s for all 54 video clips in MSR II, while Top-K search in [15] takes 26mins. This is even 2800 times faster than the branch and bound search in [7]. For the 5-hour large dataset, it only costs $37.6s$ to retrieve the top-7 results. From the statistics of Table V, we can see that the increase of database size (from 1 hour to 5 hours) do not significantly increase the computational cost (from 26.4s to 37.6s). The reason is that the search consists of two rounds: coarse search and fine search. The fine search time is almost the same for the two datasets since we only consider the similar number of candidates received from the coarse round search. On the other hand, due to the high down-sampling factor in the coarse round search, the computational burden for large dataset is not that intensive.

Note that the total computation time is independent of the duration of the query videos. This means, when there are more queries, the total computation time only grows linearly with the feature extraction time, which is around 30s for a 20s sequence. For a very large database, like Youtube, it has

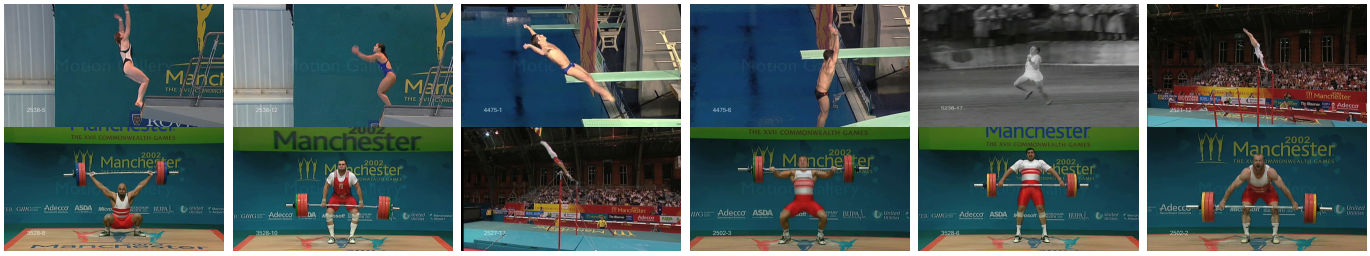

Fig. 12. Search results of diving and weightlifting on UCF Sports dataset. The first column is from the query video and the following six columns refer to the top-5 retrieved results. The fourth and fifth results for the diving action and the second result for weightlifting are false positives caused by similar motion patterns.

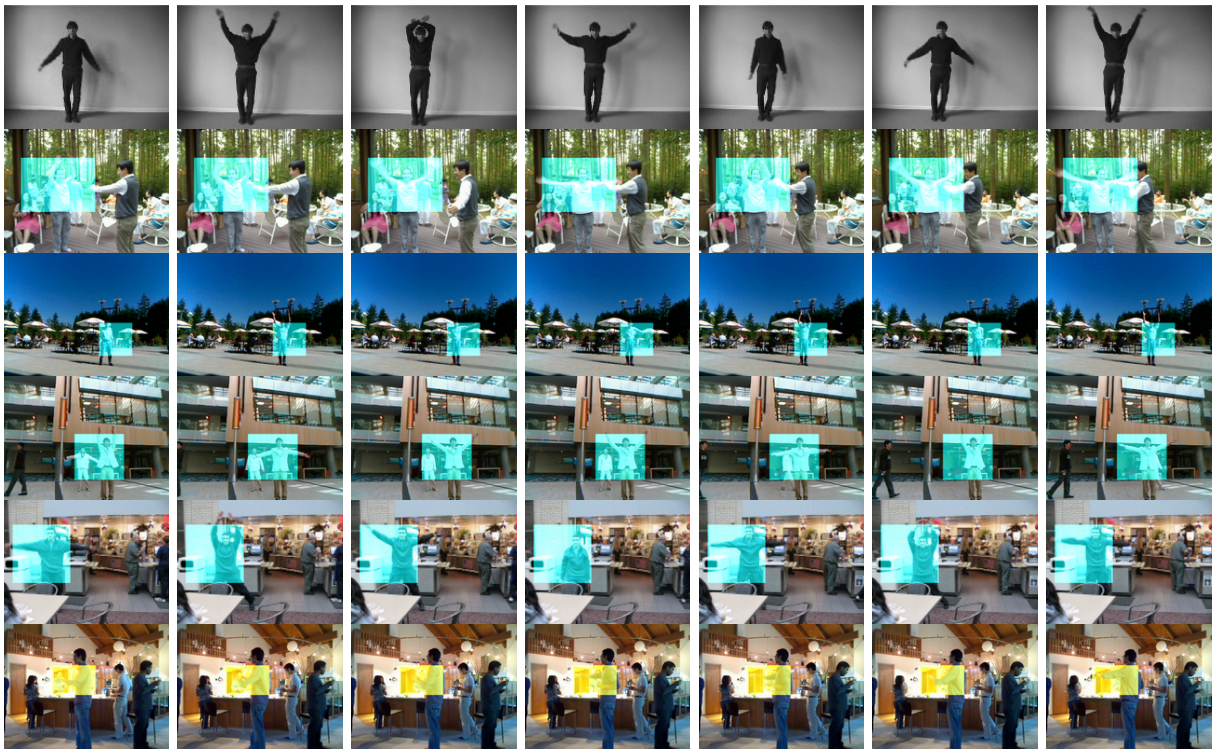

Fig. 13. Illustrations for handwaving retrieval in large dataset. First row shows the seven frames from query video while the following rows give the five retrieved examples (four positive examples marked by cyan color and one negative example marked by yellow color).

little impact to our voting cost since the voting cost mainly depends on the number of trees and the depth of each tree. In order to deal with the increasing search complexity, parallel computing can be utilized in the first step of branch and bound search since the search for different video clips are mutually independent. As the number of candidates for search in the second step of our branch and bound search only depends on the number of retrieved results required by the user, the database size has little impact on the runtime cost for the second step.

Method	Voting Time (ms)	One sequence (s)
LSH [7]	173.48±423.71	173.48
random indexing trees	0.537±0.14	0.537

TABLE IV
CPU TIME CONSUMED BY STIP VOTING IN MSR II DATABASE THAT CONSISTS OF 870,000 STIPS. THE SECOND COLUMN IS THE CPU TIME FOR COMPUTING THE VOTES OF ALL STIPS IN THE DATABASE WITH RESPECT TO A SINGLE STIP IN THE QUERY. THE THIRD COLUMN IS THE CPU TIME FOR COMPUTING THE VOTES WITH RESPECT TO A 10-SECONDS LONG QUERY VIDEO (APPROXIMATELY 1000 STIPS).

VII. CONCLUSION

We developed a fast action search system that can efficiently locate similar action instances to a query action. To index the video interest points for fast matching, we proposed to build multiple randomized-visual-vocabularies by using random indexing trees. Compared with using a single vocabulary tree, multiple vocabulary trees better compensate for information loss due to quantization. By increasing the number of vocabularies, we can improve the matching thus lead to better search accuracy. To achieve faster response time, we developed a coarse-to-fine subvolume search scheme which results in a dramatic speedup over the existing video

Fig. 14. Illustrations for handclapping retrieval in large dataset. First row shows the seven frames from query video while the following rows give the five retrieved examples (four positive examples marked by cyan color and one negative example marked by yellow color).

Dataset	MSR II (1h)	Large Dataset (5h)
Voting time (s)	0.6	0.6
Search time (s)	24.1	37
Refinement time (s)	2	0
Total Computation Time (s)	26.7	37.6

TABLE V
TOTAL COMPUTATION TIME OF OUR RETRIEVAL SYSTEM. SUPPOSE THE QUERY VIDEO IS AROUND 15S AND WE TEST IT ON TWO DATABASE: MSRII AND 5-HOUR LARGE DATASET. OUR ALGORITHM RETRIEVES THE TOP 7 SUBVOLUMES FROM THE DATABASE AS SHOWN IN FIG. 4 AND FIG. 10.

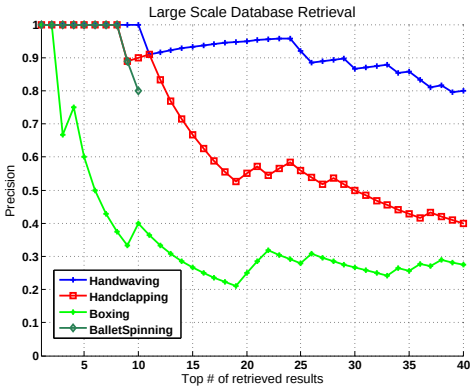

Fig. 17. Retrieval results from large scale dataset.

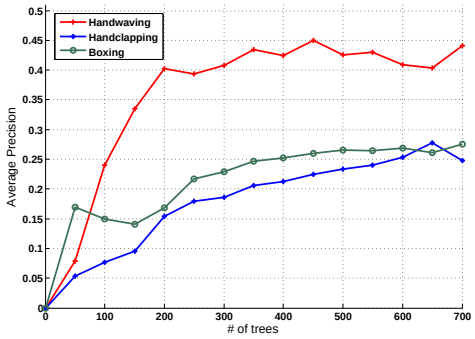

Fig. 18. Relation between number of trees and average precision.

branch-and-bound method. Various challenging cross-dataset experiments demonstrate that our proposed method is not only fast to search large-scale video dataset, but also robust to action variations, partial occlusions, and cluttered and dynamic backgrounds. Moreover, our technique has the unique property that it is easy to leverage feedbacks from the user.

REFERENCES

[1] I. Laptev, "On space-time interest points," *International Journal of Computer Vision*, vol. 64, no. 2-3, pp. 107–123, 2005.
[2] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Proc. IEEE Conf. on Pattern Recognition*, 2004.
[3] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Proc. IEEE Conf. on CVPR*, 2008.
[4] J. Gall, and V. Lempitsky, "Class-specific Hough forests for object detection," in *Proc. IEEE Conf. on CVPR*, 2009.
[5] K. Reddy, J. Liu, and M. Shah, "Incremental action recognition using feature-tree," in *Proc. IEEE Intl. Conf. on Computer Vision*, 2009.
[6] Y. Ke, R. Sukthankar, and M. Hebert, "Event detection in crowded videos," in *Proc. IEEE International Conf. on Computer Vision*, 2007.

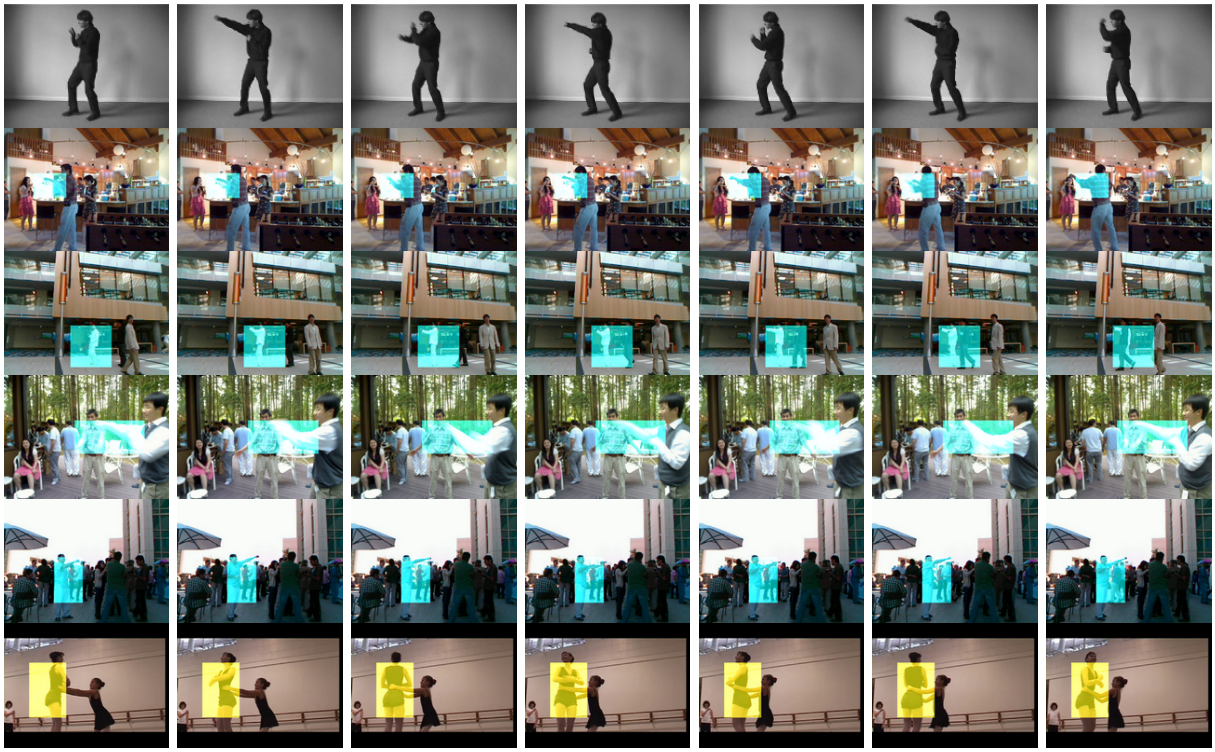

Fig. 15. Illustrations for boxing retrieval in large dataset. First row shows the seven frames from query video while the following rows give the five retrieved examples (four positive examples marked by cyan color and one negative example marked by yellow color).

[7] J. Yuan, Z. Liu, and Y. Wu, "Discriminative Video Pattern Search for Efficient Action Detection," in *IEEE Trans. on Pattern Analysis and Machine Intelligence* , 2011.

[8] J. Yuan, Z. Liu, Y. Wu, and Z. Zhang., "Speeding up spatio-temporal sliding-window search for efficient event detection in crowded videos," in *ACM Multimeida Workshop on Events in Multimedia*, 2009.

[9] D. Mikel, J. Ahmed, and M. Shah, "Action mach a spatio-temporal maximum average correlation height filter for action recognition," in *Proc. IEEE Conf. on CVPR*, 2008.

[10] Y. Ke, R. Sukthankar, and M. Hebert, " Volumetric Features for Video Event Detection," in *International journal of computer vision*, Vol. 88, pp. 339–362, 2010.

[11] P. Wang, G.D. Abowd and J.M. Rehg, "Quasi-periodic event analysis for social game retrieval," in *Intl. Conf. on Computer Vision*, 2009.

[12] Z. Jiang, Z. Lin, and L. S. Davis, "Recognizing Human Actions by Learning and Matching Shape-Motion Prototype Trees," in *Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, 2011.

[13] J. Collomosse, and G. McNeill, and Y. Qian, "Storyboard sketches for content based video retrieval," in *IEEE Intl. Conf. on Computer Vision*, 2009.

[14] L. Cao, Z. Liu and T. Huang, "Cross-dataset Action Detection," in *Proc. IEEE Proc. CVPR*, 2010.

[15] G. Yu, A. Norberto, J. Yuan, Z. Liu, "Fast Action Detection via Discriminative Random Forest Voting and Top-K Subvolume Search," in *Proc. IEEE Trans. on Multimedia*, Vol. 13, pp. 507-517, 2011.

[16] K. Derpanis, M. Sizintsev, K. Cannons, and R.P. Wildes, "Efficient action spotting based on a spacetime oriented structure representation," in *Computer Vision and Pattern Recognition*, 2010.

[17] L. Breiman, "Random forests," in *Machine learning*, Vol.45, pp.5-32, 2001.

[18] K. Mikolajczyk, and H. Uemura, "Action Recognition with Appearance-Motion Features and Fast Search Trees"," in *Computer Vision and Image Understanding*, Vol. 115, pp. 426-438, 2010.

[19] I. Laptev and P. Prez, "Retrieving actions in movies," in *Proc. IEEE Intl. Conf. on Computer Vision (ICCV)*, 2007.

[20] A. Yao, J. Gall and L. V. Gool, "A Hough Transform-Based Voting Framework for Action Recognition"," in *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[21] J. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised learning of human action categories using spatial-temporal words," in *International Journal of Computer Vision*, Vol.3, pp.299-318, 2008.

[22] Y. Aytar, M. Shah, J. Luo, "Utilizing semantic word similarity measures for video retrieval," in *Computer Vision and Pattern Recognition*, 2008.

[23] A. Gaidon, M. Marszalek and C. Schmid, "Mining visual actions from movies" in *British Machine Vision Conference*, 2009.

[24] N. Ikizler-Cinbis, R.G. Cinbis, and S. Sclaroff, "Learning actions from the web," in *Proc. IEEE Intl. Conf. on Computer Vision*, 2009.

[25] A. Kovashka and K. Grauman, "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[26] C. Lampert, M.B. Blaschko, T. Hofmann, "Efficient Subwindow Search: A Branch and Bound Framework for Object Localization," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.31, pp.2129-2142, 2009.

[27] C. Marsala, and M. Detyniecki, "High scale video mining with forests of fuzzy decision trees," in *Proc. Intl. Conf. on Soft computing as transdisciplinary science and technology*, 2008.

[28] H. Ning, T.X. Han, D.B. Walther, M. Liu, T.S. Huang, "Hierarchical space-time model enabling efficient search for human actions," in *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 19, pp. 808-820, 2009.

[29] A. Oikonomopoulos, I. Patras, M. Pantic, "Spatiotemporal Localization and Categorization of Human Actions in Unsegmented Image Sequences," in *IEEE Trans. on Image Processing*, Vol. 20, pp. 1126-1140, 2011.

[30] K. Prabhakar, S. Oh, P. Wang, G.D. Abowd, J.M Rehg, "Temporal causality for the analysis of visual events," in *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[31] J. Sun, X. Wu, S. Yan, L.F. Cheong, T.S. Chua, J. Li, "Hierarchical spatio-temporal context modeling for action recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[32] J. Liu, J. Luo, M. Shah, "Recognizing realistic actions from videos "in the wild"," in *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[33] R. Messing, C. Pal, H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," in *Proc. IEEE Intl. Conf. on Computer Vision*, 2009.

[34] L. Duan, D. Xu, I.W. Tsang, J. Luo, "Visual event recognition in videos by learning from web data"," in *Computer Vision and Pattern Recognition (CVPR)*, 2010.

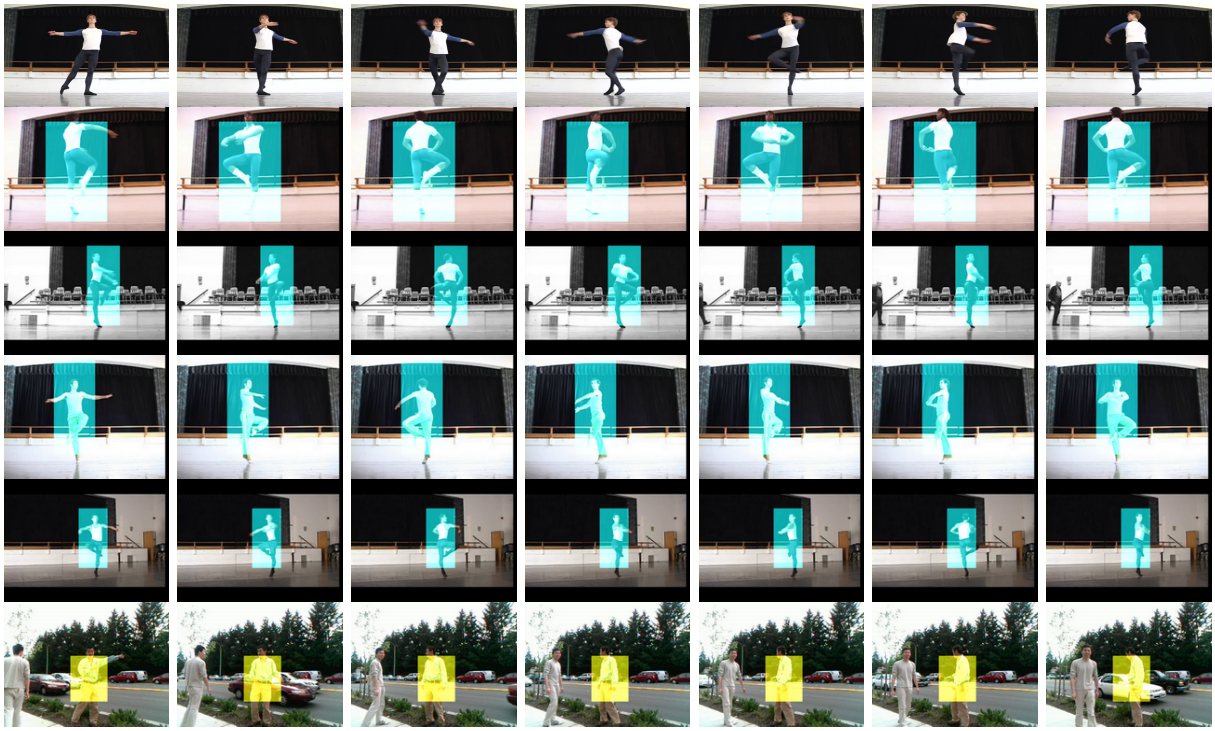

Fig. 16. Illustrations for ballet spin retrieval in large dataset. First row shows the seven frames from query video while the following rows give the five retrieved examples (four positive examples marked by cyan color and one negative example marked by yellow color).

[35] J. Niebles, C.W. Chen, F.F., Li, "Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification"," in *European Conference on Computer Vision (ECCV)*, 2010.

[36] S. Ali, A. Basharat, M. Shah, "Chaotic invariants for human action recognition," in *Proc. IEEE Intl. Conf. on Computer Vision (ICCV)*, 2007.

[37] D. Ramanan, DA Forsyth, "Automatic annotation of everyday movements," in *Proc. Neural Information Processing Systems (NIPS)*, 2003

[38] T. Kim, R. Cipolla, "Canonical correlation analysis of video volume tensors for action categorization and detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 1415-1428, 2009.

[39] H. Seo, P. Milanfar, "Action recognition from one example," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2010.

[40] G. Zhu, M. Yang, K. Yu, W. Xu, Y. Gong, "Detecting video events based on action recognition in complex scenes using spatio-temporal descriptor," in *Proc. ACM International Conference on Multimedia*, pp.165-174, Oct.19-24, 2009.

[41] E. Boyer D. Weinland, R. Ronfard, "Free viewpoint action recognition using motion history volumes," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 207–229, 2006.

[42] Y. Hu, L.L. Cao, F. Lv, S. Yan, Y. Gong, T.S. Huang, "Action detection in complex scenes with spatial and temporal ambiguities," in *Proc. IEEE Intl. Conf. on Computer Vision*, 2009.

[43] F. Bobick, J.W. Davis, "The recognition of human movement using temporal templates," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, no. 3, pp. 257–267, 2001.

[44] G. Yu, J. Yuan, Z. Liu, "Unsupervised Random Forest Indexing for Fast Action Search," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2011.

[45] L. Gorelick, M. Blank, E. Shechtman, M. Irani, R. Basri, "Actions as Space-Time Shapes," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 29, pp. 2247-2253, 2007.

[46] S. Oh, A. Hoogs, A.G.Amitha Perera, etc "A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2011.

[47] M. Marszałek, I. Laptev, C. Schmid, "Actions in Context," in *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[48] R. Ji, H. Yao, X. Sun, S. Liu, "Actor-independent action search using spatiotemporal vocabulary with appearance hashing," in *Pattern Recognition*, Vol. 44, pp. 624-638, 2011.

[49] J. Bentley, "ultidimensional binary search trees used for associative searching," in *Communications of the ACM*, Vol. 18, pp. 509–517, 1975.

[50] E. Shechtman, M. Irani, "Space-time behavior-based correlation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2045–2056, 2007.

[51] A. Efros, A. Berg, G. Mori, J. Malik, "Recognizing Action at a Distance," in *International Conference on Computer Vision*, pp 726-733, 2003.

[52] Y. Xie, H. Chang, Z. Li, L. Liang, X. Chen, D. Zhao, "A Unified Framework for Locating and Recognizing Human Actions," in *Computer Vision and Pattern Recognition*, 2011.

[53] K. Schindler, L. Van Gool, "Action snippets: How many frames does human action recognition require?," in *Computer Vision and Pattern Recognition*, 2008.

[54] R. Minhas, A.A. Mohammed, Q.M. jonathan Wu, "Incremental Learning in Human Action Recognition Based on Snippets," in *IEEE Transaction on CSVT*, 2011.