Real-time Human Action Search using Random Forest based Hough Voting*

Gang Yu, Junsong Yuan
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore
gyu1@e.ntu.edu.sg, jsyuan@ntu.edu.sg

Zicheng Liu
Microsoft Research
Redmond, WA, USA
zliu@microsoft.com

ABSTRACT

Many existing techniques in content based video retrieval treat a video sequence as a whole to match it against a query video or to assign a text label. Such an approach has serious limitations when applied to human action retrieval because an action may occur only in a sub-region and last for a small portion of the video length. In situations like this, we essentially need to match the subvolumes of the video sequences against the query video. A naive exhaustive search is impractical due to large number of possible subvolumes for each video sequence. In this paper, we propose a novel framework for action retrieval which performs pattern matching at subvolume level and is very efficient in handling large corpus of videos. We construct an unsupervised random forest to index the video database, generate a score volume with Hough voting and then employ a max sub-path strategy to quickly search for the temporal and spatial positions of all the video sequences in the database. We present action search experiments on challenging datasets to validate the efficiency and effectiveness of our system.

Categories and Subject Descriptors

H.3.3 [[**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms, Experimentation, Theory

Keywords

Action Search, Hough Voting, Random Forest Indexing, Max Sub-path Search

1. INTRODUCTION

Human action search aims to search similar action segments from large database given a query action. It has a lot of practical uses in our daily lives, e.g., human computer interface, video surveillance, and medical diagnosis. However, this topic has been much less exploited than human action classification [3] and action detection [2] [9] [7].

[8] proposed a framework for action search which has promising experimental results on some existing benchmark datasets. With the help of random forest indexing, the problem that the limited query samples cannot model the large intra-class variations can be alleviated. Besides, by coarse-to-fine branch and bound search, the algorithm [8] can finish a one-hour database search in 26 seconds. Despite great speed advantage and superior performance, the algorithm [8] has two major drawbacks. First, it is still not acceptable to search one action in 26 seconds. For online applications, fast response is the inevitable property otherwise the user experiences would suffer. Second, there is one implicit assumption in [8]: the actions should be spatially fixed. For instance, it cannot handle the moving actions, e.g., walking. This is primarily due to the sliding window (branch and bound search) approach. If we relax the search space to include the moving volumes, it would be extremely computational prohibited.

To handle the two challenges in [8], we propose a novel action search system that can search and crop similar actions from a large corpus of video clips given an exemplar video clip. Videos are represented with spatio-temporal interest points [3]. To support fast search, we build a random forest to index all the STIP points in the database. Given a query video, we match each STIP in the query with the database based on the random forest. Given the matches, Hough voting is performed to generate a 3D score volume. The value of each element in the score volume refers to the likelihood of the region belonging to the same action type as query. By searching a sub-path within the volume, we can obtain an optimal sub-volume which is the most similar to the query action. There are three major contributions of our paper.

- We propose to use Hough voting to vote for candidate action centers, which integrates the spatial structure of interest points.
- The search problem is solved with the max sub-path search algorithm rather than the branch and bound search in [8] so that we can handle the moving actions.
- According to the experiments, we algorithm outperforms the state-of-the-art method [8] on both the accuracy and the speed.

Experiments on the benchmark MSR II [9] database and a drinking database validate the efficiency and effectiveness of our algorithm.

*Area chair: Qi Tian

MM'11, November 28–December 1, 2011, Scottsdale, Arizona, USA.

2. ALGORITHM

Our action search system mainly constitutes two components, Hough voting and max sub-path searching. Given a video database, $\mathcal{D} = \{\mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_N\}$, where $\mathcal{V}_i$ refers to a video clip and N is the number of videos. Each video can be represented by a set of spatio-temporal interest points (STIP) [3], $\mathcal{V} = \{d_i\}$. For each STIP point $d_i = [f_i, l_i]$, f_i refers to the feature description and l_i refers to its spatial-temporal location, $l_i = [\mathbf{x}_i, t_i]$. We use HOG (Histogram of Gradient) and HOF (Histogram of Flow) to describe f_i with feature dimension of 162. For action retrieval, our objective is to crop all the sub-volumes which are similar to the given query video $\mathcal{Q}$.

For simplicity, we denote $\mathcal{D} = \{d_i, i = 1, 2, \cdots, N_D\}$ as the database, where N_D is the total number of STIP points. Given a query $\mathcal{Q} = \{d_q, q = 1, 2, \cdots, N_Q\}$, we want to crop the sub-volumes with the maximum similarity score:

$$S(V^*, \mathcal{Q}) = max_{t\in[a,b],\mathbf{x},\rho} s(V_{t\in[a,b]}(\mathbf{x}, \rho), \mathcal{Q}), \quad (1)$$

where $V_{t\in[a,b]}(\mathbf{x}, \rho)$ refers to the sub-volume with temporal duration from frame a to frame b and spatial center at position $\mathbf{x}$ and spatial size at ρth level. For simplicity, we assume the spatial scale (ρ will be omitted in later discussion so that only $s(V_{t\in[a,b]}(\mathbf{x}), \mathcal{Q})$ is considered) and the spatial center does not change over time in the returned sub-volume. In the later discussion, we will show that our algorithm can be easily extended to handle different spatial scales and moving actions.

2.1 Random Forest Indexing

To index the database for fast STIP matches, we employ the unsupervised random forest structure [8]. In the off-line phase, we construct N_t trees independently and each tree is a partition over the feature space. Once a new query STIP arrives, we consider all the STIPs from the database which fall on the same leaf as the query STIP, and use those STIPs as the matches of the query STIP. There are several benefits of random forest based indexing. First of all, the trees are constructed in an unsupervised way so that they are class-independent. This is of great significance for action retrieval since we cannot determine ahead of time what kind of action a user is going to retrieve. Second, random forest based indexing is fast to evaluate. As shown in [8], it is over 300 times faster than the LSH (Local Sensitive Hashing)-accelerated nearest neighbor search scheme. Third, by adding more trees in the forest, we can make our matches more accurate. The outliers caused by the trees have little impact due to the robustness of Hough voting which will be discussed in the next section.

2.2 Hough Voting

To define the similarity function, we employ the frame based Hough voting score [4]:

$$\begin{aligned} s(V_{t\in[a,b]}(\mathbf{x}), \mathcal{Q}) &= \textstyle\sum_{t\in[a,b]} \sum_{d_q\in\mathcal{Q}} p([\mathbf{x}, t], f_q, l_q) \\ &= \textstyle\sum_{t\in[a,b]} \sum_{d_q\in\mathcal{Q}} p([\mathbf{x}, t]|f_q, l_q) p(f_q, l_q), \end{aligned} \quad (2)$$

where f_q and l_q refer to the feature description and location of the qth STIP point in the query video, respectively. $p([\mathbf{x}, t], f_q, l_q)$ is the probability that there exists a demanded action in position $[\mathbf{x}, t]$ and a matched STIP point d_q in the query. Since it is reasonable to assume a uniform prior over f_q and l_q, we have

$$s(V_{t\in[a,b]}(\mathbf{x}), \mathcal{Q}) \propto \sum_{t\in[a,b]} \sum_{d_q\in\mathcal{Q}} p([\mathbf{x}, t]|f_q, l_q). \quad (3)$$

In order to compute $p([\mathbf{x}, t]|f_q, l_q)$, the query STIP d_q should be first matched with the STIPs in $\mathcal{D}$. Although nearest neighbor search and its approximated algorithms, e.g. Local Sensitive Hashing (LSH), can give good match results, it is extremely time consuming. Thus, a random forest indexing is used here to improve the matching speed. Suppose we use N_t trees to partition the feature space, every query STIP d_q will fall into one of the leaves in tree T_i ($T_i \subset \mathcal{D}$) based on f_q. All the database STIPs that fall on the same leave as d_q are considered as the matches of d_q. Then we have

$$\begin{aligned} & s(V_{t\in[a,b]}(\mathbf{x}), \mathcal{Q}) \\ & \propto \sum_{t\in[a,b]} \sum_{d_q\in\mathcal{Q}} \textstyle\sum_i p([\mathbf{x}, t]|T_i, f_q, l_q) p(T_i|f_q, l_q) \\ & = \sum_{t\in[a,b]} \sum_{d_q\in\mathcal{Q}} \textstyle\sum_i \sum_{d_j\in\mathcal{D}, d_j\in T_i} p([\mathbf{x}, t], d_j|T_i, f_q, l_q) p(T_i|f_q, l_q), \end{aligned} \quad (4)$$

where $p(T_i|f_q, l_q) = \frac{1}{N_t}$. Denote N_{T_i} to be the number of points on the same leaf as d_q in tree T_i. For point j, denote its spatio-temporal location as $[\mathbf{x}^j_{(T_i)}, t^j_{(T_i)}], j = 1, 2, \cdots, N_{T_i}$. Then $p([\mathbf{x}, t]|T_i, f_q, l_q)$ can be computed with kernel density estimation:

$$p([\mathbf{x}, t]|T_i, f_q, l_q) = \frac{1}{Z} \sum_{j=1}^{N_{T_i}} exp(-\frac{||[\hat{\mathbf{x}}^j_q - \mathbf{x}, \hat{t}^j_q - t]||^2}{\sigma^2}), \quad (5)$$

where Z is a normalization constant and σ^2 is a bandwidth parameter. Suppose $l_q = [\mathbf{x}_q, t_q]$. Denote c_q to be the spatial center position of query action. Consider the video sequence from which the jth STIP point is extracted at position $[\mathbf{x}^j_{(T_i)}, t^j_{(T_i)}]$. This point casts a vote for the action position in this video sequence. The voted action position, $[\hat{\mathbf{x}}^j_q, \hat{t}^j_q]$, is computed as

$$\begin{aligned} \hat{\mathbf{x}}^j_q &= \mathbf{x}^j_{(T_i)} - \rho(\mathbf{x}_q - c_q) \\ \hat{t}^j_q &= t^j_{(T_i)}. \end{aligned} \quad (6)$$

By setting different ρ, we can handle action sub-volumes with different scales.

Although Hough voting is widely used in image analysis, it is much less exploited in the action retrieval literature. Our Hough voting strategy has the following advantages. First, its computational complexity is much lower than that of branch and bound search. Suppose the number of trees in a forest is N_t, the tree depth is t_d, the total number of STIPs in the database is N_D, and the number of STIPs in the query video is N_Q. Let us assume the STIPs in the database are unformly distributed over different leaves of each tree. Then the computational complexity of our Hough voting step is in average $O(N_t t_d N_Q + N_t \lceil \frac{N_D}{2^{t_d}} \rceil N_Q)$. In comparison, the complexity of branch and bound is at least proportional to the size of the spacetime volume, which is much larger than the number of interest points N_Q. As shown in Table. 2, the computational cost for Hough voting step is very small. Second, the Hough voting algorithm integrates the spatial context of different STIP into consideration. This enables us to have a more accurate search results compared with point based branch and bound search [7]. Third, the Hough voting step can be seamlessly integrated into our action search

framework. In [7] Hough voting is utilized to refine the spatial locations of detections in the optional refine step. This is different from our work where Hough voting is to vote for the candidate action centers.

2.3 Sub-volume Search

After Hough voting, we have a 3D score volume (if we want to handle spatial scale changes of actions, it should be 4D score volume). What we need to do is to locate the temporal location a, b and spatial center $\mathbf{x}$ in Eq. 2. This problem is traditionally known as *Maximum Subarray Problem* [1]. Since the elements of our score volume are always non-negative, we constrain the size of the sub-volume by adding a regularization term. The score function Eq. 4 is further defined as:

$$\hat{s}(V_{t\in[a,b]}(\mathbf{x}), \mathcal{Q}) = s(V_{t\in[a,b]}(\mathbf{x}), \mathcal{Q}) + (b-a+1)\lambda S, \quad (7)$$

where λ is a small negative constant and S is spatial size of the action. Alg. 1 gives an illustration of our algorithm. We perform 1D sub-array search over different spatial position and locate a subarray with the maximum summation (Line 1 - 15). Based on the maximum summation, we can back-travel to locate the sub-path (Line 16 - 19). From the searched subarray, we can get the location of the retrieved sub-volume, i.e. a, b and $\mathbf{x}$.

In order to handle multiple scale action search, we can simply extend 3D score volume to 4D score volume with several scale levels and add another loop over different scales. Suppose our database is of spatial extent $M \times N$ and temporal duration T and further assume we consider the number of scale levels as S, the computational complexity of our sub-volume search is $O(MNTS)$. In our experiments below, S is fixed as 3. Hence, our sub-volume search is superior to branch and bound search in [7, 9] whose worst complexity is $O(M^2N^2T)$.

So far, we have assumed that the spatial position of an action does not change over time. We can relax this assumption by using the method of [6] to handle the moving actions, e.g., walking.

2.4 Implementation details

Since speed is an important aspect for action search system, we employ several methods to reduce the computational cost. Similar to [7], we down-sample the 3D score volume, both spatially and temporally. The difference between [7] and our Hough voting framework is that in [7], an interest point votes for any sub-volume that contains this point while in our framework, an interest point votes for a particular sub-volume whose center is specified. In other words, the relative spatial information is thrown away in [7] while we take advantage of the relative spatial information.

By spatial down-sampling of score volume, it is possible for us to handle high-resolution videos, e.g. 720p and 1080p, which are quite popular nowadays. Besides, with the help of temporal down-sampling, we can deal with extremely large datasets. In our experiments, we set the spatial-downsampling factor as 8 and temporal-downsampling factor as 2. As shown in Table. 1 and Table. 2, our action search system is superior to [8] not only on the precision but also on the speed.

3. EXPERIMENTS

Although action retrieval is extensively exploited before, action search with localization is little discussed in the literature. In order to evaluate our system, we compare with the recent work [8] and action detection [2]. Since they reported quantitative experiment results on MSR II dataset [9], it is easier for us to compare our results with theirs.

Algorithm 1 Maximum Sub-array Search

Input:
3D score volume $F \in \mathbb{R}^{M\times N\times T}$
Output:
Location of sub-volume $V_{t\in[a,b]}(\mathbf{z})$, i.e. $a, b, \mathbf{z} = (x, y)$

```
1: for i = 1 to M do
2:   for j = 1 to N do
3:     Set c := 0, g := −inf
4:     for t = 1 to T do
5:       Set c := c + F(i, j, t)
6:       if c < 0 then
7:         Set c := 0
8:       end if
9:       if c > g then
10:        Set g := c
11:        Set b := t, x := i, y := j
12:      end if
13:    end for
14:  end for
15: end for
16: Set a := b
17: while a > 1 and g > 0 do
18:   Set g := g − F(i, j, t), a := a − 1
19: end while
```

3.1 Experiments on MSRII

To give a quantitative results of our action retrieval system, MSR II is employed as a database for search. Three actions, handclapping, handwaving and boxing, are tested. The query videos are randomly drawn from KTH [5]. The evaluation measure is the same as [8]. To compute the 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: $\frac{\text{Volume}(V\cap G)}{\text{Volume}(V)} > \frac{1}{8}$.

Three algorithms are evaluated. They are (i) branch & bound action retrieval [8], (ii) cross-dataset action detection [2] and (iii) our Hough voting based retrieval system. Since (ii) is a detection algorithm, it utilized all the available data (around 256 video clips) for training. Average precisions for handclapping, handwaving and boxing actions are listed in Table. 1. In order to handle boxing action with different directions, (i) flipped the query video and provide both clips as query. However, in our system (iii), we do not flip the query video, and only use the original clip as the query. According to the quantitative results, our algorithm outperforms the state-of-the-art action retrieval system. Due to the space limitation, we only give one illustrating example of handwaving action with top-7 retrieved results in Fig. 1. The regions marked with blue color indicate the retrieved spatial location of the action.

3.2 Experiments on wild videos

Another illustrating example is shown here with a wild video database. The database is downloaded from the Youtube

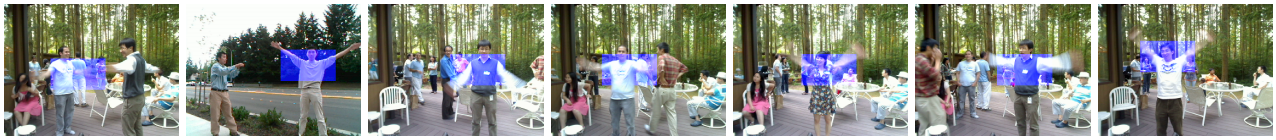

Figure 1: Top-7 retrieved results of handwaving action from MSR II database.

Figure 2: Action retrieval results on a Hollywood movie segment. A three second long query video with drinking action is shown in the first column where the images on the three rows in this column are its sample frames. The second to fifth column list the top-4 retrieved results, respectively.

Action Type	B&B Search [8]	Action Detection [2]	Our Method
handclapping	0.2397	0.1316	**0.3609**
handwaving	0.4301	0.3671	**0.5415**
boxing	0.3029	0.1748	**0.3166**

Table 1: Average precisions for action retrieval and action detection on MSR II database. The second and fourth columns are for action retrieval. The third column is for action detection.

Method	B&B Search [8]	Our Method
Voting time (s)	0.6	1.5
Search time (s)	24.1	0.8
Total Time (s)	24.7	2.3

Table 2: Comparison of the total computation time of two action retrieval systems. The query video is 20 seconds long and the database consists of 54 high resolution videos with a total length of one hour. The task is to retrieve the 7 best matches from the database.

and comes from one part of the movie called *"Coffee and Cigarettes"*. It contains 3415 frames and includes several drinking actions. We choose a query video with drinking action around 3 seconds long and 99 frames. Fig. 2 shows the top 4 retrieved results. The first column are the query video where the three rows in the column are the example frames from the video. The second to fifth columns are the top 4 retrieved videos. The second column and the fifth column are false detections. The reason for the false detections is that the motions in the two instances are similar to the drinking motion.

3.3 Computational Cost

Table. 2 shows the total computation cost (CPU time only) for branch and bound search system [8] and our system. We use a single PC with 2.6GHz CPU and 3G memory, which is almost the same environment as in [8]. We can see that our max sub-path search algorithm is 30 times faster than branch and bound action search. Overall, our system is over 10 times faster than [8].

4. CONCLUSION

We proposed an action search system which is superior to the state-of-the-art methods in terms of both the precision and the computational cost. By indexing the video database with a random forest, our frame-based Hough voting obtains a 3D score volume very efficiently. The computational cost can be further reduced by spatial and temporal down-sampling of the score volume. With the max sub-path search, our algorithm quickly detects the spatio-temporal positions of the action instances in the video database. The experiments validate the effectiveness and efficiency of our proposed system.

5. REFERENCES

[1] J. Bentley. Programming pearls: algorithm design techniques. *Communications of the ACM*, 27(9):865–873, 1984.

[2] L. Cao, Z. Liu, and T. S. Huang. Cross-dataset action detection. In *CVPR*, pages 1998–2005, 2010.

[3] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2):107–123, 2005.

[4] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, 2008.

[5] C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR (3)*, pages 32–36, 2004.

[6] D. Tran and J. Yuan. Optimal Spatio-Temporal Path Discovery for Video Event Detection. *IEEE CVPR*, 2011.

[7] G. Yu, N. Goussies, J. Yuan, and Z. Liu. Fast action detection via discriminative random forest voting and top-k subvolume search. *Multimedia, IEEE Transactions on*, 13(3):507–517, 2011.

[8] G. Yu, J. Yuan, and Z. Liu. Unsupervised Random Forest Indexing for Fast Action Search. *IEEE CVPR*, 2011.

[9] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. *IEEE CVPR*, 2009.