Discriminative Subvolume Search for Efficient Action Detection

Junsong Yuan
EECS Dept., Northwestern Univ.
Evanston, IL, USA
j-yuan@u.northwestern.edu

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

Ying Wu
EECS Dept., Northwestern Univ.
Evanston, IL, USA
yingwu@eecs.northwestern.edu

Abstract

*Actions are spatio-temporal patterns which can be characterized by collections of spatio-temporal invariant features. Detection of actions is to find the re-occurrences (*e.g.* through pattern matching) of such spatio-temporal patterns. This paper addresses two critical issues in pattern matching-based action detection: (1) efficiency of pattern search in 3D videos and (2) tolerance of intra-pattern variations of actions. Our contributions are two-fold. First, we propose a discriminative pattern matching called naive-Bayes based mutual information maximization (NBMIM) for multi-class action categorization. It improves the state-of-the-art results on standard KTH dataset. Second, a novel search algorithm is proposed to locate the optimal subvolume in the 3D video space for efficient action detection. Our method is purely data-driven and does not rely on object detection, tracking or background subtraction. It can well handle the intra-pattern variations of actions such as scale and speed variations, and is insensitive to dynamic and clutter backgrounds and even partial occlusions. The experiments on versatile datasets including KTH and CMU action datasets demonstrate the effectiveness and efficiency of our method.*

1. Introduction

Actions can be treated as spatio-temporal objects which are characterized as 3-dimensional volumetric data. Like the use of sliding windows in object detection, action detection in a video can be formulated as locating 3D subvolumes that contain the target action. Despite previous successes of sliding window-based object detection [10], locating desired actions in the video space is still a challenging problem, mainly due to the following two difficulties.

First, searching for actions in the video space is much more complicated than searching for objects in the image space. Without knowing the location, temporal duration, and the spatial scale of the action, the search space for video actions is prohibitive for exhaustive search. For example, a one-minute video sequence of size $160 \times 120 \times 1800$ contains more than 10^{14} spatial-temporal subvolumes of various sizes and locations. This number is more than 10^6 times

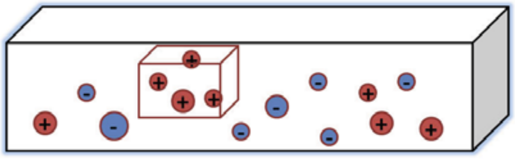

Figure 1. Action detection is formulated as searching for a subvolume in video that has the maximum mutual information toward the action class. Each circle represents a spatio-temporal feature point which contributes a vote based on its own mutual information.

larger than the number of bounding boxes that an image of size 160×120 can generate. Therefore, although some state-of-the-art approaches can efficiently search the 2D image space for object detection [10], they are not scalable to search 3D videos, due to the enormous search space. To reduce such a huge search space, some other methods try to avoid exhaustive search by sampling the search space, *e.g.* only considering a fixed number of spatial and temporal scales [9]. However, this treatment is under the risk of missing detections. Moreover, after subsampling, the solution space is still very large.

Second, human actions involve tremendous intra-pattern variations. The same type of actions can be completely different in their visual appearances, due to the variations in performing speed, clothing, scale, view points, not to mention partial occlusion. When using a single and rigid action template for pattern matching [9] [4], the actions that vary from the template can not be detected. One potential remedy is to use multiple templates to cover more variations, but the required number of templates increases rapidly resulting in formidable computational costs.

We propose an efficient action detection approach that addresses the two challenges mentioned above. As illustrated in Fig. 1, a video sequence is represented by a collection of spatio-temporal invariant points (STIPs), where each STIP casts a positive or negative-valued vote for the action class, based on its mutual information w.r.t. the action class. Action detection can then be formulated as the problem of searching for the 3D subvolume that has maximum total votes. Such a 3D subvolume has a maximum mutual information w.r.t. the action class. This is a new formulation for action detection. To handle the large search space in 3D video, our proposed method decouples the temporal and spatial spaces and applies different search strategies on them to speed up the search. In addition, to make an analogy to the template-based pattern matching, our discriminative matching can be regarded as the use of two template classes, one from the entire positive training data and the other from the negative samples, based on which, discriminative learning is exploited for more accurate pattern matching.

The benefits of our method are three-fold. First, the proposed discriminative pattern matching can well handle action variations by using all of the training data instead of a single template. By incorporating the negative training information, our pattern matching has better discriminative power across different action classes. Second, unlike conventional action detection methods which require object tracking and detection, our method is a pure data-driven approach that does not rely on object tracking or detection. In the meanwhile, as our method does not depend on background subtraction, it can tolerate clutter and moving backgrounds. Last but not least, the proposed search method for 3D videos is computationally efficient and is suitable for a real time system implementation. The experiments on various action data sets including cross-action data validate the effectiveness and efficiency of our method.

2. Related Work

Action categorization and detection has been an active research topic and many methods have been proposed. One type of approaches uses motion trajectories to represent actions and it requires target tracking [16] [1].Another type of approaches uses sequences of silhouettes or body contours to model actions [8] [14] and it requires background subtraction. Contemporary methods for action categorization use local spatio-temporal features to characterize the video and perform classification over the set of local features [13] [12] [17] [19] To improve the classification performance, both shape and motion information are applied for action categorization [15] [21] [22]. In terms of representing actions, some methods characterize an action as a spatio-temporal template, such as motion history [4] and space-time shapes [3], so that action classification and detection can be done through finding the matches of the spatio-temporal template [20] [9] [18] [7].

Some recent work in object detection were related to our proposed method as well. In [10], object detection was formulated as finding the optimal bounding box that gives the highest detection score in the image. An efficient branch-and-bound method was proposed to search for the optimal bounding box in the image. In [5], a naive-Bayesian nearest neighbor classifier was proposed. Based on the "query-to-class" distance, it achieves very good results in object categorization and largely improves the performance of nearest neighbor classifier based on the "query-to-image" distance.

3. Action Model and Matching

3.1. "Bag of Features" Representation for Actions

We represent an action as a space-time object and characterize it by a collection of spatio-temporal interest points (STIPs) [11]. Compared with the SIFT feature in the 2D image domain, STIP is an extension of invariant features to 3D video data. After detecting STIPs, two types of features can be used to describe them [12]: histogram of gradient (HOG) and histogram of flow (HOF), where HOG is the appearance feature and HOF is the motion feature. As STIPs are locally invariant for the 3D video, such features are relatively robust to action variations due to the changes in performing speed, scale, lighting condition and cloth. We denote a video sequence by $\mathcal{V} = \{\mathbf{I}_t\}$, where each frame $\mathbf{I_t}$ consists of a collection of STIPs, $\mathbf{I_t} = \{d_i\}$. We do not select key-frames but collect all STIPs to represent a video clip by $\mathcal{Q} = \{d_i\}$.

3.2. Discriminative Matching

We denote by $d \in \mathbb{R}^N$ a feature vector describing a STIP; $\mathbf{C} = \{1, 2, ..., C\}$ is the class label set. Based on the naive Bayes assumption and by assuming the independence among the STIPs, we can evaluate the mutual information between a video clip $\mathcal{Q}$ and a specific class $c \in \mathbf{C}$ as:

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

where $s^c(d_q) = MI(\mathbf{C} = c, d_q)$ is the mutual information score for d_q w.r.t. class c. The final decision of $\mathcal{Q}$ is based on the summation of the mutual information from all primitive features $d_q \in \mathcal{Q}$ w.r.t. class c. To evaluate the contribution $s^c(d_q)$ of each $d_q \in \mathcal{Q}$, we calculate the mutual information through discriminative learning:

$$\begin{aligned}
& s^c(d_q) = MI(\mathbf{C} = c, d_q) = \log \frac{P(d_q|\mathbf{C} = c)}{P(d_q)} \\
= \;& \log \frac{P(d_q|\mathbf{C} = c)}{P(d_q|\mathbf{C} = c)P(\mathbf{C} = c) + P(d_q|\mathbf{C} \neq c)P(\mathbf{C} \neq c)} \\
= \;& \log \frac{1}{P(\mathbf{C} = c) + \frac{P(d_q|\mathbf{C} \neq c)}{P(d_q|\mathbf{C} = c)} P(\mathbf{C} \neq c)}. \qquad (1)
\end{aligned}$$

Assuming an equal prior, *i.e.* $P(\mathbf{C} = c) = \frac{1}{C}$, we have:

$$s^c(d_q) = \log \frac{C}{1 + \frac{P(d_q|\mathbf{C} \neq c)}{P(d_q|\mathbf{C} = c)}(C - 1)}. \qquad (2)$$

From Eq. 2, we can see that the likelihood ratio test $\frac{P(d_q|\mathbf{C} \neq c)}{P(d_q|\mathbf{C} = c)}$ determines whether d_q votes positively or negatively for class c. When $MI(\mathbf{C} = c, d_q) > 0$, *i.e.* likelihood ratio $\frac{P(d_q|\mathbf{C} \neq c)}{P(d_q|\mathbf{C} = c)} > 1$, d_q votes a positive score $s^c(d_q)$

for the class c. Otherwise if $MI(\mathbf{C} = c, d_q) \leq 0$, *i.e.* $\frac{P(d_q|\mathbf{C} \neq c)}{P(d_q|\mathbf{C}=c)} \leq 1$, d_q votes a negative score for the class c. After receiving the votes from every $d_q \in \mathcal{Q}$, we can make the final classification decision for $\mathcal{Q}$ based on its mutual information toward C classes.

For the C-class action categorization, we built C one-against-all classifiers. The test action $\mathcal{Q}$ is classified as the class that gives the largest detection score.

$$c^* = \arg\max_{c \in \{1,2,..,C\}} MI(c, \mathcal{Q}) = \arg\max_{c \in \{1,2,..,C\}} \sum_{d \in \mathcal{Q}} s^c(d).$$

We call this naive-Bayes based mutual information maximization (NBMIM).

3.3. Computing the Likelihood Ratio

Denote by $\mathbb{T}^{c+} = \{\mathcal{V}_i\}$ the positive training dataset of class c, where $\mathcal{V}_i \in \mathbb{T}^{c+}$ is a video of class c. As each $\mathcal{V}$ is characterized by a collection of STIPs, we represent the positive training data by the collection of all positive STIPs: $\mathbb{T}^{c+} = \{d_j\}$. Symmetrically, the negative data is denoted by $\mathbb{T}^{c-}$, which is the collection of all negative STIPs.

To evaluate the likelihood ratio for each $d \in \mathcal{Q}$, we apply the kernel density estimation based on the training data $\mathbb{T}^{c+}$ and $\mathbb{T}^{c-}$. With a Gaussian kernel $K(\cdot)$ and by using the nearest neighbor approximation as in [5], we have the likelihood ratio:

$$\begin{aligned}\frac{P(d|\mathbf{C} \neq c)}{P(d|\mathbf{C} = c)} &= \frac{\frac{1}{|\mathbb{T}^{c-}|}\sum_{d_j \in \mathbb{T}^{c-}} K(d - d_j)}{\frac{1}{|\mathbb{T}^{c+}|}\sum_{d_j \in \mathbb{T}^{c+}} K(d - d_j)} \\ &\approx \lambda^c \exp^{-\frac{1}{2\sigma^2}(\|d - d_{NN}^{c-}\|^2 - \|d - d_{NN}^{c+}\|^2)} \quad (3)\end{aligned}$$

Here d_{NN}^{c-} and d_{NN}^{c+} are the nearest neighbors of d in class $c-$ and $c+$, respectively, and $\lambda^c = \frac{|\mathbb{T}^{c+}|}{|\mathbb{T}^{c-}|}$.

Adaptive Kernel Bandwidth:
For a Gaussian kernel, it is important to use appropriate kernel bandwidth σ in density estimation. A large kernel bandwidth may over smooth the density function while a too small kernel bandwidth only uses the nearest neighbor for the final result. Instead of using a fixed kernel as in [5], we propose an adaptive kernel strategy, which adjusts the kernel bandwidth based on the purity in the neighborhood of a STIP. For a $d \in \mathcal{Q}$, we denote its ϵ-nearest neighbors in class c by $NN_\epsilon^{c+}(d) = \{d_j \in \mathbb{T}^{c+} : \|d_j - d\| \leq \epsilon\}$. Correspondingly we denote by $NN_\epsilon(d) = \{d_j \in \mathbb{T}^{c+} \cup \mathbb{T}^{c-} : \|d_j - d\| \leq \epsilon\}$ the whole ϵ-nearest neighbors of d.

We now define the ϵ-purity of d by $w_\epsilon(d) = \frac{|NN_\epsilon^{c+}(d)|}{|NN_\epsilon(d)|}$. As $NN_\epsilon^{c+}(d) \subseteq NN_\epsilon(d)$, we have $w_\epsilon(d) \in [0, 1]$. To adaptively adjust the kernel size, we choose $2\sigma^2 = \frac{1}{w_\epsilon(d)}$. Denote by $\gamma(d) = \|d - d_{NN}^{c-}\|^2 - \|d - d_{NN}^{c+}\|^2$. Based on Eq. 2, the adjusted voting score for each STIP for class c is:

$$s^c(d) = \log \frac{C}{1 + \lambda^c \exp^{-\gamma(d) w_\epsilon(d)}(C-1)}. \quad (4)$$

Essentially, $w_\epsilon(d)$ describes the purity of the class c in the ϵ-NN of point d. The larger the $w_\epsilon(d)$, the more reliable the prediction it gives, and thus the stronger the voting score $s^c(d)$. In the special case when d is an isolated point such that $|NN_\epsilon^{c+}(d)| = |NN_\epsilon(d)| = 0$, we treat it as a noise point and set $w_\epsilon(d) = 0$. Thus it does not contribute any vote to the final decision as $s^c(d) = 0$ according to Eq. 4.

Efficient Nearest Neighbor Search
For every STIP $d \in \mathcal{Q}$, we need to search for its nearest neighbors in order to obtain the voting score $s^c(d)$. Therefore a number of nearest neighbor queries need to be performed depending on the size of $|\mathcal{Q}|$. To improve the efficiency of searching for nearest neighbors in the high-dimensional feature space, we apply locality sensitive hashing for the approximate ϵ-NN search [6].

4. Action Detection in Video

4.1. Subvolume Mutual Information Maximization

The task of action detection is to identify where (spatial location in the image) and when (temporal location) the action occurs in the video. Based on our NBMIM criterion, we give a new formulation of action detection as a subvolume mutual information maximization problem. Given a video sequence $\mathcal{V}$, the goal is to find a spatial-temporal subvolume (3D subvolume) $V^* \subset \mathcal{V}$, such that it has the maximum mutual information on class c:

$$\begin{aligned} V^* &= \arg\max_{V \subseteq \mathcal{V}} MI(V, \mathbf{C} = c) \quad (5) \\ &= \arg\max_{V \subseteq \mathcal{V}} \sum_{d \in V} s^c(d) = \arg\max_{V \in \mathbf{\Lambda}} f(V), \end{aligned}$$

where $f(V) = \sum_{d \in V} s^c(d)$ is the objective function and $\mathbf{\Lambda}$ denotes the candidate set of all valid 3D subvolumes in $\mathcal{V}$. Suppose the target video $\mathcal{V}$ is of size $m \times n \times t$. The optimal solution $V^* = t^* \times b^* \times l^* \times r^* \times s^* \times e^*$ has 6 parameters to be determined, where $t^*, b^* \in [0, m]$ denote the top and bottom positions, $l^*, r^* \in [0, n]$ denote the left and right positions, and $s^*, e^* \in [0, t]$ denote the start and end positions. As a counterpart of the bounding-box based object detection, the solution V^* is the 3D bounding volume that has the highest score for the target action.

The total number of the 3D subvolumes is in the order of $O(n^2m^2t^2)$. Therefore, it is computationally prohibitive to perform an exhaustive search to find the optimal subvolume V^* from such an enormous candidate pool. In the following, we first present the naive 3D branch-and-bound solution extended from 2D bounding-box search in [10], and then present our new method to search V^* more efficiently.

4.2. Efficient Search for the Optimal 3D Subvolume

4.2.1 Naive 3D branch-and-bound

A branch-and-bound solution is proposed in [10] for searching the optimal bounding box in an image for object detection. This idea can be extended to find the optimal 3D subvolume in videos. Denote by $\mathbb{V}$ a collection of 3D subvolumes. Assume there exist two subvolumes V_{min} and V_{max} such that for any $V \in \mathbb{V}$, $V_{min} \subseteq V \subseteq V_{max}$. Then we have $f(V) \leq f^+(V_{max}) + f^-(V_{min})$, where $f^+(V) = \sum_{d \in V} \max(s^c(d), 0)$ contains only positive votes, while $f^-(V) = \sum_{d \in V} \min(s^c(d), 0)$ contains only negative ones. We denote the upper bound of $f(V)$ for all $V \in \mathbb{V}$ by:

$$\hat{f}(\mathbb{V}) = f^+(V_{max}) + f^-(V_{min}) \geq f(V). \qquad (6)$$

With this upper bound, we get a straightforward extension of [10], with replacement of the 2D bounding box by a 3D subvolume. In order to distinguish this method from our new method, we call it as the naive 3D branch-and-bound method. Compared to the 2D bounding box searching, the search of 3D subvolume is much more difficult. In 3D videos, the search space has two additional parameters (start and end on the time dimension) and increases from 4-dimensions to 6-dimensions. As the complexity of the branch-and-bound grows exponentially in the number of dimensions, the naive branch-and-bound solution is too slow for 3D videos.

4.2.2 Our new efficient search

Instead of directly applying branch-and-bound in the 6D parameter space, our new method decomposes it into two subspaces: (1) 4D spatial parameter space and (2) 2D temporal parameter space. We denote by $W \in \mathbb{R}^2 \times \mathbb{R}^2$ a spatial window and $T \in \mathbb{R} \times \mathbb{R}$ a temporal segment. A 3D subvolume V is uniquely determined by W and T. The detection score of a subvolume $f(V_{W \times T})$ is: $f(V_{W \times T}) = f(W, T) = \sum_{d \in W \times T} s(d)$. Let $\mathbb{W} = [0, m] \times [0, n]$ be the parameter space of the spatial windows, and $\mathbb{T} = [0, t]$ be the parameter space of temporal segments. Our objective here is to find the spatio-temporal subvolume which has the maximum detection score:

$$[W^*, T^*] = \arg \max_{W \subseteq \mathbb{W}, T \subseteq \mathbb{T}} f(W, T). \qquad (7)$$

We take different search strategies in the two subspaces $\mathbb{W}$ and $\mathbb{T}$ and search alternately between $\mathbb{W}$ and $\mathbb{T}$.

First, if the spatial window W is determined, we can easily search for the optimal temporal segment in space $\mathbb{T}$:

$$F(W) = \max_{T \subseteq \mathbb{T}} f(W, T), \qquad (8)$$

This relates to a 1D max subvector problem and we will discuss its efficient solution later.

To search the spatial parameter space $\mathbb{W}$, we employ a branch-and-bound strategy. Since the efficiency of a branch-and-bound based algorithm critically depends on the tightness of the upper bound, we first derive a tighter upper bound.

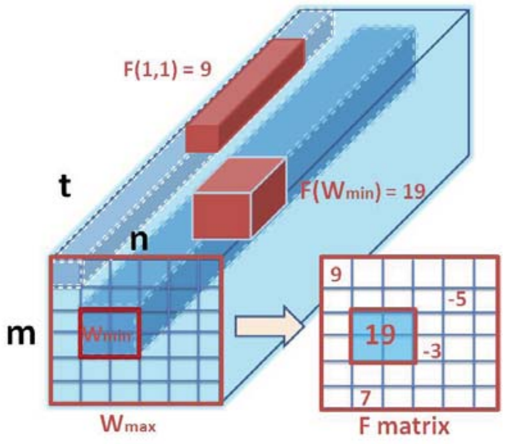

Figure 2. Illustration of the upper bound in Lemma 1. The upper bound is $\hat{F}_1(\mathbb{W}) = 19 + 9 + 7 = 35$.

Given an arbitrary parameter space $\mathbb{W} = [m_1, m_2] \times [n_1, n_2]$, we denote by $W^* = \arg\max_{W \in \mathbb{W}} F(W)$ the optimal solution, and denote by $F(\mathbb{W}) = F(W^*)$. Assume there exist two sub-rectangles W_{min} and W_{max} such that $W_{min} \subseteq W \subseteq W_{max}$ for any $W \in \mathbb{W}$. For each pixel $i \in W_{max}$, denote by $F(i) = max_{T \subseteq \mathbb{T}} f(i, T)$ the maximum sum of the 1D subvector along the temporal direction at pixel i's location. Let $F^+(i) = max(F(i), 0)$, we have the first upper bound for $F(\mathbb{W})$, illustrated in Fig. 2.

Lemma 1 *(upper bound $\hat{F}_1(\mathbb{W})$)*

$$F(\mathbb{W}) \leq \hat{F}_1(\mathbb{W}) = F(W_{min}) + \sum_{i \in W_{max}, i \notin W_{min}} F^+(i).$$

When $W_{max} = W_{min}$, we have the tight bound $\hat{F}_1(\mathbb{W}) = F(W_{min}) = F(W^*)$.

Symmetrically, for each pixel $i \in W_{max}$, denote by $G(i) = min_{T \subseteq \mathbb{T}} f(i, T)$ the minimum sum of the 1D subvector at pixel i's location. Let $G^-(i) = min(G(i), 0)$, we have the other upper bound for $F(\mathbb{W})$.

Lemma 2 *(upper bound $\hat{F}_2(\mathbb{W})$)*

$$F(\mathbb{W}) \leq \hat{F}_2(\mathbb{W}) = F(W_{max}) - \sum_{i \in W_{max}, i \notin W_{min}} G^-(i).$$

When $W_{max} = W_{min}$, we have the tight bound $\hat{F}_2(\mathbb{W}) = F(W_{max}) = F(W^*)$.

The proof of Lemma 2 is in the Appendix. As Lemma 1 and Lemma 2 are symmetric, we omit its proof due to the page limit. Based on Lemma 1 and Lemma 2, we can obtain a final tighter upper bound, which is the minimum of the two available upper bounds:

Theorem 1 *(Tighter upper bound $\hat{F}(\mathbb{W})$)*

$$F(\mathbb{W}) \leq \hat{F}(\mathbb{W}) = \min\{\hat{F}_1(\mathbb{W}), \hat{F}_2(\mathbb{W})\} \qquad (9)$$

Based on the upper bound derived in Theorem 1, we propose our new branch-and-bound solution in the spatial parameter space $\mathbb{W}$ in Alg.1. Different from the naive 3D branch-and-bound solution, the new algorithm keeps track of the current best solution which is denoted by W^* in Alg.1. Only when a parameter space $\mathbb{W}$ contains potentially better solution (*i.e.* $\hat{F}(\mathbb{W}) > F^*$), we push it into the queue. It thus avoids a waste of memory and CPU resources in maintaining the priority queue.

```
Alg.1: our new method
Require: video $\mathcal{V} \in \mathbb{R}^{m\times n\times t}$
Require: quality bounding function $\hat{F}$ (see text)
Ensure: $V^* = \arg\max_{V\subseteq\mathcal{V}} f(V)$
set $\mathbb{W} = [T, B, L, R] = [0, n] \times [0, n] \times [0, m] \times [0, m]$
get $\hat{F}(\mathbb{W}) = \min\{\hat{F}_1(\mathbb{W}), \hat{F}_2(\mathbb{W})\}$
push ($\mathbb{W}$, $\hat{F}(\mathbb{W})$) into empty priority queue $P$
set current best solution $\{W^*, F^*\} = \{W_{max}, F(W_{max})\}$;
repeat
retrieve top state $\mathbb{W}$ from $P$ based on $\hat{F}(\mathbb{W})$
if ($\hat{F}(\mathbb{W}) > F^*$)
    split $\mathbb{W} \to \mathbb{W}^1 \cup \mathbb{W}^2$
    CheckToUpdate($\mathbb{W}_1$, $W^*$, $F^*$, $P$);
    CheckToUpdate($\mathbb{W}_2$, $W^*$, $F^*$, $P$);
else
    $T^* = \arg\max_{T\subset[0,t]} f(W^*, T)$;
    return $V^* = [W^*, T^*]$.

function CheckToUpdate($\mathbb{W}$, $W^*$, $F^*$, $P$)
Get $W_{min}$ and $W_{max}$ of $\mathbb{W}$
if ($F(W_{min}) > F^*$)
    update $\{W^*, F^*\} = \{W_{min}, F(W_{min})\}$;
if ($F(W_{max}) > F^*$)
    update $\{W^*, F^*\} = \{W_{max}, F(W_{max})\}$;
if ($W_{max} \neq W_{min}$)
    get $\hat{F}(\mathbb{W}) = \min\{\hat{F}_1(\mathbb{W}), \hat{F}_2(\mathbb{W})\}$
    if $\hat{F}(\mathbb{W}) > F^*$
        push ($\mathbb{W}$, $\hat{F}(\mathbb{W})$) into $P$
```

Efficient estimation of the upper bound $\hat{F}(\mathbb{W})$
To estimate the upper bound in Theorem 1, as well as to search for the optimal temporal segment T^* given a spatial window W, we design an efficient way to evaluate $F(W_{max})$, $F(W_{min})$, and in general $F(W)$. According to Eq. 8, given a spatial window W of a fixed size, we need to search for a temporal segment with maximum summation. This problem can be formulated as the 1D max subvector problem, where given a real vector of length T, the output is the contiguous subvector of the input that has the maximum sum. The 1D max-subvector problem is a classic problem in one-dimension pattern recognition. There exists an elegant solution called Kadane's algorithm which is of a linear complexity using dynamic programming [2]. By applying the trick of integral-image, the evaluation of $F(W)$ using Kadane's algorithm can be done in a linear time. Thus the estimation of the upper bound $\hat{F}(\mathbb{W})$ is of a linear complexity $O(t)$.

	naive 3D B&B	our method
Dim. for B&B	6 (spatial-temporal)	4 (spatial)
Upper bound est.	$O(1)$	$O(t)$
Worst case	$O(m^2n^2t^2)$	$O(m^2n^2t)$

Table 1. Complexity comparison of our method and naive 3D branch-and-bound.

The worst case complexity comparison of our method and the 3D branch-and-bound is presented in Table 1. Although the upper bound estimation in our method is $O(t)$ compared with the constant complexity in naive 3D branch-and-bound, the overall complexity of our algorithm is $O(m^2n^2t)$ which is better than that of the naive 3D branch-and-bound which is $O(m^2n^2t^2)$.

5. Experiments

5.1. Action Categorization

The KTH dataset contains six types of human actions: walking, jogging, running, boxing, hand waving and hand clapping, each of which is performed several times by 25 subjects. Each video sequence exhibits one individual action. There are 4 different environments where the video sequences are captured: outdoors, outdoors with scale variation, outdoors with different clothes and indoors. We follow the standard experimental setting of KTH dataset as in [12]. Among the 25 persons, 16 of them are used for training and the rest 9 are used for testing. The training dataset contains 1528 individual actions and the testing dataset contains 863 individual actions. The video resolution is 160×120. We apply both HOG and HOF features (162-dimensional feature vector).

The categorization results are presented in Table 3. We apply the adaptive kernel size for density estimation, with $\epsilon = 2.6$ for ϵ-NN search. We set $\lambda^c = 1$ in Eq. 3 by assuming $|\mathbb{T}^{c+}| = |\mathbb{T}^{c-}|$. Among the 863 testing actions, we obtained 58 errors, and the total accuracy was 93.3%. Most of the errors are due to the mis-classification of running to jogging. In Table 2, we further compare our results to the state-of-the-art results in [12]. We apply exactly the same training and testing dataset, as well as the same STIP features. Besides using a different classifier, we do not quantize STIPs into "words". Our NBMIM-based method improves the state-of-the-art result. On one hand, this result shows the discriminative power of the STIP. On the other

	clap	wave	walk	box	run	jog
clapping	**142**	0	0	1	0	1
waving	5	**139**	0	0	0	0
walking	0	0	**144**	0	0	0
boxing	0	0	0	**143**	0	0
running	4	1	0	0	**103**	36
jogging	3	0	3	0	4	**134**

Table 3. Confusion matrix for the KTH action datasets.

	training	testing	features	classifier	accuracy
[12]	8 persons training + 8 persons cross-validation	9 persons	STIP + “bag of words”	non-linear SVM	91.8 %
ours	16 persons	9 persons	STIP	NBMIM	93.3 %

Table 2. Comparison with the state-of-the-art result on KTH dataset.

kernel size	NBMIM	NBNN
fixed ($\epsilon = 2.0$)	92.2%	91.8%
adaptive ($\epsilon = 2.0$)	93.0%	N.A.
fixed ($\epsilon = 2.6$)	92.6%	92.7%
adaptive ($\epsilon = 2.6$)	93.3%	N.A.

Table 4. Comparison of NBMIM and NBNN, with different selections of ϵ.

hand, it shows that it is unnecessary to quantize primitive features into “words” for classification. This is consistent with the discussion in [5] which pointed out that “bag-of-features” representation has the potential to provide better classification performance than “bag-of-words” model because the quantization step in the “bag-of-words” model results in the loss of discriminative information.

In Table 4, we compare the proposed NBMIM method with NBNN in [5], together with the comparison of different parameters of ϵ-NN search for density estimation. On the selection of the parameter ϵ for density estimation, it shows that the selection of $\epsilon = 2.6$ gives a slightly better performance than $\epsilon = 2.0$. Overall, our method is insensitive to the choice of ϵ as long as it is within a reasonable range. Moreover, the results of our NBMIM that uses adaptive kernel sizes outperforms that of the NBNN method for different selections of ϵ. It shows the improvement because the discriminative learning performs better than only using the likelihood ratio.

5.2. Action Detection

5.2.1 Efficiency comparison

To validate the efficiency gain of our method in searching the 3D videos, we compare our method (Alg.1) with the naive 3D branch-and-bound. We use the MVI-142a sequence in the CMU action dataset [9] for testing. The max subvolume is of size $43 \times 32 \times 112$. The input video $\mathcal{V}$ is of size $120 \times 160 \times 141$, a temporal segment from MVI-142a. We intentionally choose such a target video of a short length, such that the lengths of its 3 dimensions are balanced. This gives a fair comparison to the naive 3D branch-and-bound, because the longer the video length t, the less efficient the naive 3D branch-and-bound will be.

Fig. 3 shows that our proposed method converges much faster than the naive 3D branch-and-bound. In terms of the number of branches, our method converges after $10,302$ branches, an order of magnitude faster than the naive 3D branch-and-bound which needs $103,202$ branches before convergence. This validates that the upper bound proposed in Theorem 1 is much tighter than that of the naive 3D branch-and-bound.

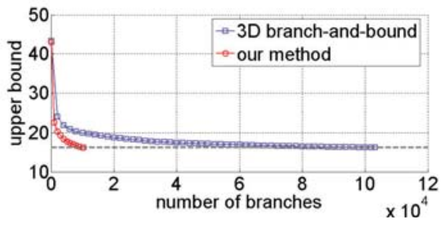

Figure 3. Comparison of the convergence.

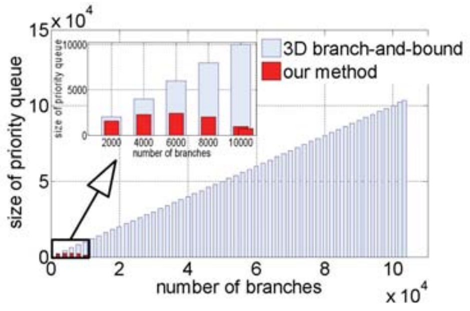

Figure 4. Comparison of the size of the priority queue.

As mentioned earlier, another advantage of our method is through keeping track of the current best solution. Only when the upper bound is better than the current best solution, we push it into the queue. In comparison, the method proposed in [10] needs to push every middle state into the priority queue, as there is no record of the current best solution. In Fig. 4, we compare the required size of the priority queue between our method and the naive 3D branch-and-bound. The size of the priority queue in our method is well controlled and is much smaller. In our method, during the branch-and-bound process, the size of the priority queue decreases after a peak value. However, for the naive 3D branch-and-bound, the size of priority queue always increases, almost linearly to the number of branches. Since each insertion or extraction operation of priority queue is $O(logn)$ for a queue of size n, the size of the priority queue affects both the computational and memory costs. It is especially important to limit a queue to a moderate size for the 3D video space search because it can generate a much larger number of candidates than the 2D image case.

5.2.2 Detecting two-hand waving action

We select the two-hand waving action as a concrete example for action detection. To validate the generalization ability of our method, we apply completely different dataset for training (KTH dataset) and testing (CMU action dataset [9]). As summarized in Table 5, for the positive training data, we apply the standard KTH hand waving dataset which contains 16 persons. The negative training data is constituted by two

pos. train	hand-waving 16 persons (KTH)
neg. train	walking 16 persons (KTH) + 1 indoor seq.
test	two-hand waving + jumping jags (CMU)

Table 5. Experimental setup of two-hand waving detection. The training and testing are from different datasets.

parts (1) the standard KTH walking dataset which contains 16 persons and (2) one office indoor sequences which contains typical actions such as sitting down and standing up. The testing dataset contains 48 sequences, which includes two classes in the CMU dataset: (1) two-hand waving and (2) jumping jags, as both of these two classes contain two-hand waving actions. The duration of each test sequence is from 10 to 40 seconds, with a resolution 160×120. Among the 48 sequences, 19 of them contain a total number of 52 positive instances. The other 29 sequences do not contain positive examples.

Considering that only a very small portion of pixels $d \in \mathcal{V}$ correspond to STIP and have non-zero voting scores, we put a constant negative prior $s(d) = -5 \times 10^{-5}$ to the rest of zero pixels. With such a negative prior, we put a penalty on detections of large spatio-temporal scales. To evaluate the results, we apply similar measurement proposed in [9] but with a loose criterion. For the precision score, detection is regarded as correct if at least $1/8$ of the volume size overlaps with the ground truth label. For the recall score, ground truth is regarded as retrieved if at least $1/8$ of its volume size is covered by at least one detection.

For multiple instance detection in the same target video sequence, we detect the first instance (subvolume with maximum mutual information) and check if its detection score is larger than the detection threshold. If it is a valid detection, we clear it by setting all of its pixels to zero values and continue to find another subvolume of the maximum summation. This process continues until the current max subvolume is not a valid detection. Fig. 6 shows the precision-recall curve of our method, by changing the detection thresholds. When selecting an appropriate threshold, both precision and recall scores can achieve around 70%. We present some of our detection results in Fig. 5.

To further speed up the search process, we slightly modify Alg.1, to obtain a trade-off between efficiency and accuracy. During the iterations of Alg.1, if F^* is larger than a detection threshold (specified by the user), it is surely that there is a valid detection in the corresponding parameter space $\mathbb{W}$. Therefore, we speed up the search by limiting the rest of the search within $\mathbb{W}$ only. This leads to a much faster convergence. To evaluate the trade-off between efficiency and accuracy quantitatively, we tested 5 sequences from CMU action dataset (details in supplementary materials). We find that although there is no guarantee to obtain the optimal solution, the search speed is usually hundreds of times faster, and the solution is still quite close to the optimal one.

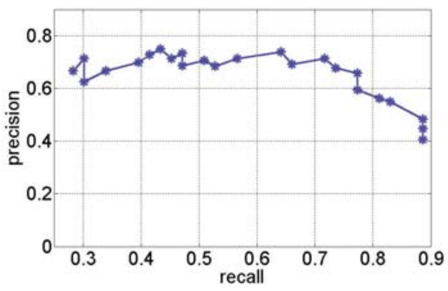

Figure 6. Performance of two-hand wave detection in CMU dataset. See texts for the definition of precision and recall.

The overall cost of our method is from three aspects: (1) extraction of STIPs; (2) calculation of voting scores and (3) search of the 3D subvolume. First, the detection of the STIPs is 4-8 frames per second, depending on the video contents. Second, by using LSH for efficient nearest neighbor search, the query time of each STIP is only 10 to 20 milliseconds. As each frame contains 5-10 STIPs on average, the processing time is 5-20 frames per second. Finally, for the efficiency purpose, we apply the approximated strategy mentioned above in searching for the 3D bounding box in our experiment. This leads to an efficient MATLAB implementation at around 5-30 frames per second. Overall, our method has the potential to be implemented in real-time.

5.2.3 Multi-class action detection

We select 3 types of actions: boxing, hand waving and hand clapping from the KTH dataset as the training set (16 persons each class). For the negative training data, we use the walking class (16 persons) from the KTH dataset, plus five indoor sequences that contain actions sitting down and standing up. For each action class, its negative training data includes both the negative class and the other two action classes. We collect 10 video sequences of length from 34 to 60 seconds. The test sequences are captured in both indoor and outdoor scenes, with clutter backgrounds. Each sequence contains multiple types of actions. The test sequences contain in total 13 hand waving, 10 hand clapping and 14 boxing actions. Among the 37 action instances, 21 of them are correctly detected and there are only 3 false detections.

6. Conclusion

We characterize a video action as a spatio-temporal point collection and apply discriminative pattern matching for action detection. Instead of using a single template, we proposed the NBMIM method that treats all the STIPs from positive training samples as an integrated template and apply both positive and negative templates for discriminative learning. Although such a model ignores the spatio-temporal dependency among features, it brings a much better tolerance of intra-pattern variations, including clothes

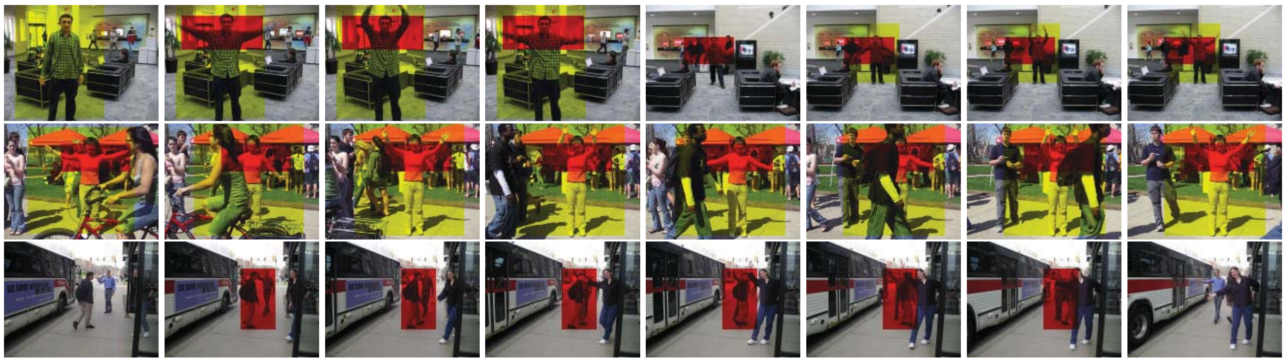

Figure 5. Detection results of two-hand waving. The yellow bounding box is the ground truth label of the whole human body action and the red bounding box is our detection of two-hand waving. The 1_{nd} row: detection under multiple spatial scales; the 2_{rd} row: detection with partial occlusions; the 3_{th} row: a false detection caused by two individual hand-wavings from two different persons. More results can be seen in the supplementary materials.

changes, performing style and speed variations of actions. Our results on standard KTH dataset show the improvement over the state-of-the-art results on action categorization.

For efficient pattern search and action detection, a novel solution is proposed to search the 3D video space. Based on a tighter upper bound, our algorithm is significantly more efficient in 3D subvolume search and thus action detection. As a data-driven approach, our method does not rely on human tracking and detection, and it can automatically handle scale variations, clutter and moving background, and even partial occlusions.

Acknowledgment

The majority of the work was carried out when Junsong Yuan was a summer intern at Microsoft Research, Redmond, Washington. We thank Dr. Cha Zhang, Dr. Yan Ke and Dr. Zhengyou Zhang for very helpful discussions. This work was supported in part by National Science Foundation grant IIS-0347877 and US Army Research Laboratory and the US Army Research Office under grant ARO W911NF-08-1-0504.

Appendix

We prove the upper bound in Lemma 2 here.

$$\begin{aligned} f(W^*, T^*) &= F(W^*) = \sum_{i \in W^* \times T^*} s(i) \\ &= \sum_{i \in W_{max} \times T^*} s(i) - \sum_{i \in (W_{max} \backslash W^*) \times T^*} s(i) \\ &\leq F(W_{max}) - \sum_{i \in (W_{max} \backslash W^*)} G(i) \\ &\leq F(W_{max}) - \sum_{i \in (W_{max} \backslash W^*)} G^-(i) \\ &\leq F(W_{max}) - \sum_{i \in (W_{max} \backslash W_{min})} G^-(i), \end{aligned}$$

where $i \in (W_1 \backslash W_2)$ denotes $i \in W_1, i \notin W_2$. When $W_{max} = W_{min}$, we have $\sum_{i \in (W_{max} \backslash W_{min})} G^-(i) = 0$, which gives the tight bound $F(W^*) = F(W_{max})$.

References

[1] S. Ali, A. Basharat, and M. Shah. Chaotic invariants for human action recognition. In *Proc. IEEE International Conf. on Computer Vision*, 2007. 2

[2] J. Bentley. Programming pearls. *Algorithm Design Techniques*, 27(9):865–871, 1984. 5

[3] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *Proc. IEEE International Conf. on Computer Vision*, 2005. 2

[4] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 23(3):257–267, 2001. 1, 2

[5] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2008. 2, 3, 6

[6] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distribution. In *Proc. of Twentieth Annual Symposium on Computational Geometry*, pages 253–262, 2004. 3

[7] A. A. Efros, A. C.Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proc. IEEE International Conf. on Computer Vision*, 2003. 2

[8] A. Galata, N. Johnson, and D. Hogg. Learning variable-length markov models of behaviour. *Computer Vision and Image Understanding (CVIU)*, 81:398–413, 2001. 2

[9] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *Proc. IEEE International Conf. on Computer Vision*, 2007. 1, 2, 6, 7

[10] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2008. 1, 2, 3, 4, 6

[11] I. Laptev. On space-time interest points. *Intl. Journal of Computer Vision*, 2005. 2

[12] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2008. 2, 5, 6

[13] J. Liu, J. Luo, and M. Shah. Feature mining and fusion strategies for recognizing realistic actions from videos in the wild. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2009. 2

[14] T. B. Moeslund, A. Hilton, and V. Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding (CVIU)*, 104:90–126, 2006. 2

[15] P. Natarajan and R. Nevatia. View and scale invariant action recognition using multiview shape-flow models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2008. 2

[16] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005. 2

[17] J. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, 2008. 2

[18] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2008. 2

[19] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proc. ACM Multimedia*, 2007. 2

[20] E. Shechtman and M. Irani. Space-time behavior based correlation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005. 2

[21] S. N. Vitaladevuni, V. Kellokumpu, and L. S. Davis. Action recognition using ballistic dynamics. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2008. 2

[22] A. Yilmaz and M. Shah. Actions as objects: a novel action representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005. 2