# Propagative Hough Voting for Human Activity Recognition

Gang Yu[1], Junsong Yuan[1], and Zicheng Liu[2]

[1] School of Electrical and Electronic Engineering, Nanyang Technological University
gyu1@e.ntu.edu.sg, jsyuan@ntu.edu.sg
[2] Microsoft Research Redmond, WA, USA
zliu@microsoft.com

**Abstract.** Hough-transform based voting has been successfully applied to both object and activity detections. However, most current Hough voting methods will suffer when insufficient training data is provided. To address this problem, we propose propagative Hough voting for activity analysis. Instead of letting local features vote individually, we perform feature voting using random projection trees (RPT) which leverage the low-dimension manifold structure to match feature points in the high-dimensional feature space. Our RPT can index the unlabeled feature points in an unsupervised way. After the trees are constructed, the label and spatial-temporal configuration information are propagated from the training samples to the testing data via RPT. The proposed activity recognition method does not rely on human detection and tracking, and can well handle the scale and intra-class variations of the activity patterns. The superior performances on two benchmarked activity datasets validate that our method outperforms the state-of-the-art techniques not only when there is sufficient training data such as in activity recognition, but also when there is limited training data such as in activity search with one query example.

## 1 Introduction

Hough-transform based local feature voting has shown promising results in both object and activity detections. It leverages the ensemble of local features, where each local feature votes individually to the hypothesis, thus can provide robust detection results even when the target object is partially occluded. Meanwhile, it takes the spatial or spatio-temporal configurations of the local features into consideration, thus can provide reliable detection in the cluttered scenes, and can well handle rotation or scale variation of the target object.

Despite previous successes, most current Hough-voting based detection approaches require sufficient training data to enable discriminative voting of local patches. For example, [7] requires sufficient labeled local features to train the random forest. When limited training examples are provided, e.g., given one or few query examples to search similar activity instances, the performance of previous methods is likely to suffer. The root of this problem lies on the challenges

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part III, LNCS 7574, pp. 693–706, 2012.

of matching local features to the training model. Due to the possibly large variations of activity patterns, if limited training examples are provided, it is difficult to tell whether a given local feature belongs to the target activity or not. Thus the inaccurate voting scores will degrade the detection performance.

In this paper, we propose *propagative Hough voting* for activity analysis. To improve the local feature point matching, we introduce random projection trees [18], which are able to capture the intrinsic low dimensional manifold structure to improve matching in high-dimensional space. With the improved matching, the voting weight for each matched feature point pair can be computed more reliably. Besides, as the number of trees grows, our propagative Hough voting algorithm is theoretically guaranteed to converge to the optimal detection.

Another nice property of the random projection tree is that its construction is unsupervised, thus making it perfectly suitable for leveraging the unlabeled test data. When the amount of training data is small such as in activity search with one or few queries, one can use the test data to construct the random projection trees. After the random projection trees are constructed, the label information in the training data can then be propagated to the testing data by the trees.

Our method is explained in Fig. 1. For each local patch (or feature) from the training example, it searches for the best matches through RPT. Once the matches in the testing video are found, the label and spatial-temporal configuration information are propagated from the training data to the testing data. The accumulated Hough voting score can be used for recognition and detection. By applying the random projection trees, our proposed method is as efficient as the existing Hough-voting based activity recognition approach, e.g., the random forest used in [7]. However, our method does not rely on human detection and tracking, and can well handle the intra-class variations of the activity patterns. With an iterative scale refinement procedure, our method can handle small scale variations of activities as well.

We evaluate our method in two benchmarked datasets, UT-interaction [10] and TV Human Interaction [19]. To fairly compare with existing methods, we test our propagative Hough voting with (1) insufficient training data, e.g., in activity search with few query examples, and (2) sufficient training data, e.g., in activity recognition with many training examples. The superior performances over the state-of-the-arts validate that our method can outperform in both conditions.

## 2 Related Work

Based on the successful development of video features, e.g., STIP [1], cuboids [13], and 3D HoG [22], many human activity recognition methods have been developed. Previously, [15][7][16][11] rely on the human detection or even human pose estimation for activity analysis. But human detection, tracking, and pose estimation in uncontrolled environments are challenging problems.

Without relying on auxiliary algorithms such as human detection, [12][5][6] perform activity recognition by formulating the problem as a template matching process. In [12], it learns a spatial-temporal graph model for each activity and classifies

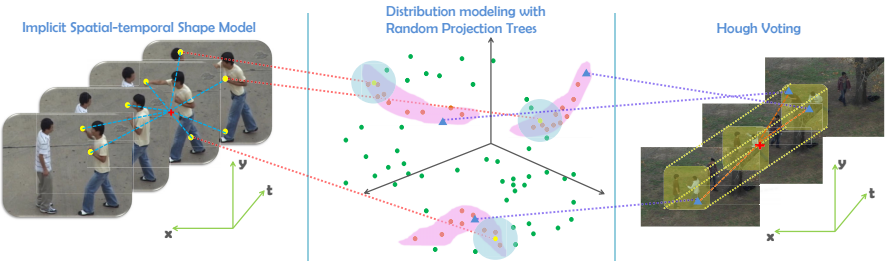


**Fig. 1.** *Propagative Hough Voting*: The left figure illustrates our implicit spatial-temporal shape model on a training video. Three sample STIPs from the testing videos are illustrated with blue triangles in the right figure. Several matches will be found for the three STIPs given the RPT. We choose three yellow dots to describe the matched STIPs from the training data in the middle figure. For each training STIP (yellow dot), the spatial-temporal information will be transferred to the matched testing STIPs (blue triangle) in the testing videos. By accumulating the votes from all the matching pairs, a sub-volume is located in the right figure. The regions marked with magenta color refer to the low-dimension manifold learned with RPT, which can built on either training data or testing data. (Best viewed in color)

the testing video as the one with the smallest matching cost. In [5], temporal information is utilized to build the "string of feature graph". Videos are segmented at a fixed interval and each segment is modeled by a feature graph. By combining the matching cost from each segment, they can determine the category of the testing video as the one with the smallest matching cost. In [6], similar idea of partitioning videos to small segments is used but video partition problem is solved with dynamic programming. There are several limitations in these matching based algorithms. First, the template matching algorithms, e.g., graph matching [12][5], are computationally intensive. For example, in order to use these template based methods for activity localization, we need to use sliding-windows to scan all the possible sub-volumes, which is an extremely large search space. Despite the fact that [6] can achieve fast speed, the proposed dynamic BoW based matching is not discriminative since it drops all the spatial information from the interest points. Similarly, in [12][5], the temporal models are not flexible to handle speed variations of the activity pattern. Third, a large training dataset will be needed to learn the activity model in [12][6]. However, in some applications such as activity search, the amount of training data is extremely limited.

To avoid enumerating all the possible sub-volumes and save the computational cost, Hough voting has been used to locate the potential candidates. In [7], Hough voting has been employed to vote for the temporal center of the activity while the spatial locations are pre-determined by human tracking. However, tracking human in unconstrained environment is a challenging problem. In contrast, our algorithm does not rely on tracking. In our algorithm, both spatial and temporal centers can be determined by Hough voting and the scale can be further refined

with back-projection. Besides, the trees in [7] are supervisedly constructed for the classification purpose while our trees are trying to model the underlying data distribution in an unsupervised way. Furthermore, our trees can be built on the test data which allows our propagative Hough voting to well handle the limited training data problem.

## 3 Activity Recognition by Detection

Spatial-temporal interest points (STIP) [1] are first extracted from each video. For each STIP, we describe it with Histogram of Gradient and Histogram of Optical Flow. In total, the feature dimension is 162. We represent each training video with implicit spatial-temporal shape model based on extracted STIP points as shown in Fig. 1. Although we only apply sparse STIP feature in our experiments, our method is also applicable to dense local features. We refer to the training data as $\mathcal{R} : \{d_r = [\mathbf{f}_r, l_r]; r = 1, 2, \cdots, N_{\mathcal{R}}\}$, where $\mathbf{f}_r$ and $l_r$ are the descriptor and 3D location of interest point $d_r$, respectively. $N_{\mathcal{R}}$ is the number of interest points.

Suppose we have a set of testing videos, denoted by $\mathcal{S} = \{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_{N_{\mathcal{S}}}\}$, we want to recognize and locate the specific activity in the training video set $\mathcal{R}$, where $\mathcal{R}$ can be one or more training examples. Our goal is to find a video sub-volume, $V^*$, to maximize the following similarity function:

$$max_V S_{V \subset \mathcal{S}}(V, \mathcal{R}) = max_{\mathbf{x},t,\rho} S(V(\mathbf{x}, t, \rho), \mathcal{R}), \tag{1}$$

where $V(\mathbf{x}, t, \rho)$ refers to the sub-volume with temporal center $t$ and spatial center $\mathbf{x}$; $\rho$ refers to the scale size and duration; $S(\cdot, \cdot)$ is the similarity measure. In total, we have 6 parameters (center position x, y, t, and width, height, duration) to locate the optimal sub-volume $V^*$. For the problem of multi-class activity recognition (suppose we have K classes), Eq. 1 will be searched K times with training data $\mathcal{R}$ from different categories. Later, the class with the highest score of $V^*$ will be picked as the predicting label for the testing video $\mathcal{V}$.

To measure the similarity between $V(\mathbf{x}, t, \rho)$ and the training data $\mathcal{R}$, similar to [17], we define $S(V(\mathbf{x}, t, \rho), \mathcal{R})$ in Eq. 1 as follow.

$$\begin{aligned} S(V(\mathbf{x}, t, \rho), \mathcal{R}) &= \textstyle\sum_{d_r \in \mathcal{R}} p([\mathbf{x}, t, \rho], d_r) \\ &= \textstyle\sum_{d_r \in \mathcal{R}} p([\mathbf{x}, t, \rho] | d_r) p(d_r), \end{aligned} \tag{2}$$

where $d_r = [\mathbf{f}_r, l_r]$ with $\mathbf{f}_r$ representing the feature description and $l_r$ representing the location of the $r$th STIP point in the training videos. $p([\mathbf{x}, t, \rho], \mathbf{f}_r, l_r)$ is the probability that there exists a target activity at position $[\mathbf{x}, t, \rho]$ and a matched STIP point $d_r$ in the training data. Since it is reasonable to assume a uniform prior over $d_r$, we skip $p(d_r)$ and focus on the local feature voting $p([\mathbf{x}, t, \rho] | d_r)$:

$$\begin{aligned} p([\mathbf{x}, t, \rho] | d_r) &= \textstyle\sum_{d_s \in \mathcal{S}} p([\mathbf{x}, t, \rho], d_s | d_r) \\ &= \textstyle\sum_{d_s \in \mathcal{S}} p([\mathbf{x}, t, \rho] | d_s, d_r) p(d_s | d_r) \\ &= \textstyle\sum_{d_s \in \mathcal{S}} p([\mathbf{x}, t, \rho] | l_s, l_r) p(\mathbf{f}_s | \mathbf{f}_r). \end{aligned} \tag{3}$$

In Eq. 3, $p(\mathbf{f}_s | \mathbf{f}_r)$ determines the voting weight which relies on the similarity between $\mathbf{f}_s$ and $\mathbf{f}_r$. We will elaborate on how to compute $p(\mathbf{f}_s | \mathbf{f}_r)$ in Section 4.

On the other hand, $p([\mathbf{x}, t, \rho]|l_s, l_r)$ determines the voting position. Suppose $d_r = [\mathbf{f}_r, l_r] \in \mathcal{R}$ matches $d_s = [\mathbf{f}_s, l_s] \in \mathcal{S}$, we cast the spatial-temporal information from the training data to the testing data with voting position $l_v = [\mathbf{x}_v, t_v]$:

$$\begin{aligned} \mathbf{x}_v &= \mathbf{x}_s - \eta_{\mathbf{x}}(\mathbf{x}_r - c_r^{\mathbf{x}}) \\ t_v &= t_s - \eta_t(t_r - c_r^t), \end{aligned} \tag{4}$$

where $[\mathbf{x}_s, t_s] = l_s$, $[\mathbf{x}_r, t_r] = l_r$, $[c_r^{\mathbf{x}}, c_r^t]$ is the spatio-temporal center position of the training activity and $\eta = [\eta_{\mathbf{x}}, \eta_t]$ refers to the scale level and duration level (the scale size of the testing video, i.e., $\rho$, over the scale size of the matched training video).

Once the voting position for testing sequence is available, we can compute $p([\mathbf{x}, t, \rho]|l_s, l_r)$ as:

$$p([\mathbf{x}, t, \rho]|l_s, l_r) = \frac{1}{Z} e^{-\frac{||[\mathbf{x}_v - \mathbf{x}, t_v - t]||^2}{\sigma^2}}, \tag{5}$$

where $Z$ is a normalization constant and $\sigma^2$ is a bandwidth parameter.

## 4 Propagative Interest Point Matching

The matching of local features $p(\mathbf{f}_s|\mathbf{f}_r)$ plays an essential role in our Hough voting. According to Eq. 3, as each $d_r \in \mathcal{R}$ will be matched against all $d_s \in \mathcal{S}$, an efficient and accurate matching is essential. We propose to use the random projection trees [18] (RPT), which is constructed in an unsupervised way, to model the underlying low-dimension feature distribution, as the light magenta regions shown in Fig. 1. Compared with traditional Euclidean distance which ignores the hidden data distribution, RPT can give a more accurate evaluation of $p(\mathbf{f}_s|\mathbf{f}_r)$ with the help of underlying data distribution.

RPT has three unique benefits compared with other data structures, e.g., [21]. First of all, as proven in [18], random projection trees can adapt to the low-dimension manifold existing in a high dimension feature space. Thus, the matching found by random projection trees is superior to the nearest neighbor based on Euclidean distance. This advantage is further validated by our experimental results in Section 6. Second, similar to BoW model, we quantize the feature space by tree structures. Rather than enumerating all the possible interest point matches, we can efficiently find the matches by passing the query interest point from the root to the leaf nodes. This can save a lot of computational cost. Third, we can make more accurate estimation by increasing the number of trees. Later, we will prove that our random projection tree based Hough voting generates optimal solution when the number of trees approaches infinity. In the following section, we describe how to implement the random projection trees.

### 4.1 Random Projection Trees

Depending on the applications, our random projection trees can be built on (1) training data only, e.g., standard action classification and detection, (2) testing

data only, e.g., activity search, (3) both training and testing data. The trees are constructed in an unsupervised way and the labels from the training data will only be used in the voting step. Assume we have a set of STIPs, denoted by $\mathcal{D} = \{d_i; i = 1, 2, \cdots, N_{\mathcal{D}}\}$, where $d_i = [\mathbf{f}_i, l_i]$ as defined in Section 3 and $N_{\mathcal{D}}$ is the total number of interest points. The feature dimension is set to $n = 162$, so $\mathbf{f}_i \in R^n$.

**Algorithm 1.** Trees = *ConstructRPT*($\mathcal{D}$)

1: **for** $i = 1 \rightarrow N_T$ **do**
2: BuildTree($\mathcal{D}$, 0)
3: **end for**

4: **Proc** Tree = BuildTree($\mathcal{D}$, depth)
5: **if** depth $< \delta_d$ **then**
6: Choose a random unit direction $v \in R^n$
7: Pick any $x \in \mathcal{D}$; find the farthest point $y \in \mathcal{D}$ from $x$
8: Choose $\gamma$ uniformly at random in $[-1, 1] \cdot 6||x - y||/\sqrt{n}$
9: Rule(x) := $x \cdot v \leq (median(\{z \cdot v; z \in \mathcal{D}\}) + \gamma)$
10: LTree $\leftarrow$ BuildTree($\{x \in \mathcal{D}; Rule(x) = true\}$, depth+1)
11: RTree $\leftarrow$ BuildTree($\{x \in \mathcal{D}; Rule(x) = false\}$, depth+1)
12: **end if**

We implement random projection trees [18] as shown in Algorithm 1. There are two parameters related to the construction of trees. $N_T$ is the number of trees and $\delta_d$ is the maximum tree depth. Each tree can be considered as one partition of the feature space to index the interest points.

At the matching step, $p(\mathbf{f}_s|\mathbf{f}_r)$ in Eq. 3 will be computed as:

$$p(\mathbf{f}_s|\mathbf{f}_r) = \frac{1}{N_T} \sum_{i=1}^{N_T} I_i(\mathbf{f}_s, \mathbf{f}_r), \tag{6}$$

where $N_T$ refers to the number of trees and

$$I_i(\mathbf{f}_s, \mathbf{f}_r) = \begin{cases} 1, & \mathbf{f}_s, \mathbf{f}_r \text{ belong to the same leaf in tree } T_i \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Thus, Eq. 2 becomes

$$\begin{aligned} S(V(\mathbf{x}, t, \rho), \mathcal{R}) &\propto \sum_{d_r \in \mathcal{R}} \sum_{i=1}^{N_T} \sum_{d_s \in \mathcal{S}} I_i(\mathbf{f}_s, \mathbf{f}_r) p([\mathbf{x}, t, \rho]|l_s, l_r) \\ &\propto \sum_{d_r \in \mathcal{R}} \sum_{i=1}^{N_T} \sum_{d_s \in \mathcal{S} \ \&\& \ I_i(\mathbf{f}_s, \mathbf{f}_r)=1} p([\mathbf{x}, t, \rho]|l_s, l_r), \end{aligned} \tag{8}$$

where $d_s \in \mathcal{S}$ && $I_i(\mathbf{f}_s, \mathbf{f}_r) = 1$ refers to the interest points from $\mathcal{S}$ which fall in the same leaf as $d_r$ in the $ith$ tree. Based on Eq. 5, we can compute the voting score as:

$$S(V(\mathbf{x}, t, \rho), \mathcal{R}) \propto \sum_{d_r \in \mathcal{R}} \sum_{i=1}^{N_T} \sum_{d_s \in \mathcal{S} \ \&\& \ I_i(\mathbf{f}_s, \mathbf{f}_r)=1} e^{-\frac{|||[\mathbf{x}_v - \mathbf{x}, t_v - t]|||^2}{\sigma^2}}. \tag{9}$$

### 4.2 Theoretical Justification

The matching quality of Eq. 6 depends on the number of trees $N_T$. To justify the correctness of using random projection trees for interest point matching, we show that, when the number of trees is sufficient, our Hough voting algorithm can obtain the optimal detection results. For simplicity, we assume our hypothesis space is of size $W \times H \times T$, with $W, H, T$ refer to the width, height and duration of the testing data, respectively. Each element refers to a possible center position for one activity and the scale $\rho$ is fixed. We further assume there is only one target activity existing in the search space at the position $l^* = [\mathbf{x}^*, t^*]$. So in total there are $N_H = W \times H \times T - 1$ background positions. To further simplify the problem, we only vote for one position for each match rather than a smoothed region in Eq. 5. That is,

$$p(l^*|l_s, l_r) = \begin{cases} 1, l^* = l_v \\ 0, otherwise \end{cases} \tag{10}$$

We introduce a random variable $\mathbf{x}^{(i)}$ with Bernoulli distribution to indicate whether we have a vote for the position $l^*$ or not in the $ith$ match. We refer to the match accuracy as $q$ and therefore $p(\mathbf{x}^{(i)} = 1) = q$. We introduce another random variable with Bernoulli distribution $\mathbf{y}^{(i)}$ to indicate whether we have a vote for the background position $l_j$ (where $l_j \neq l^*$) or not in the $ith$ match. Suppose each background position has an equal probability to be voted, then $p(\mathbf{y}^{(i)} = 1) = \frac{1-q}{N_H}$. We prove the following theorem in the supplementary material.

**Theorem 1.** ***Asymptotic property of propagative Hough voting:*** *When the number of trees $N_T \to \infty$, we have $S(V(l^*), \mathcal{R}) > S(V(l_j), \mathcal{R})$ with probability $1 - \Phi(\frac{-(q - \frac{1-q}{N_H})\sqrt{N_M}}{\sigma_{xy}})$. Specifically, if $q \geq \frac{1}{N_H+1}$ , we have $S(V(l^*), \mathcal{R}) > S(V(l_j), \mathcal{R})$ when the number of trees $N_T \to \infty$.*

In Theorem 1, $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{x^2}{2}} dx$ and $\sigma_{\mathbf{xy}}$ refers to the variance. $N_M$ refers to the number of matches according to Eq. 8: $N_M = N_T \times N_{\mathcal{R}} \times N_L$ if we build our RPT on the testing data, and $N_M = N_T \times N_{\mathcal{S}} \times N_L$ if we build our RPT on the training data. $N_L$, referring to the average number of interest points in each leaf, can be estimated as $N_L \approx \frac{N_{\mathcal{D}}}{2^{\delta_d}}$ where $\delta_d$ denotes the tree depth and $N_{\mathcal{D}}$ the size of the data for building RPT. Based on our empirical simulation experiments, $q$ is much larger than $\frac{1}{N_H+1}$. Thus the asymptotic property is true.

## 5 Scale Determination

To estimate $\rho$ in activity localization, we propose an iterative refinement method, which iteratively applies the Hough voting and scale refinement. The reason we use the iterative algorithm is that we have 6 parameters to search for. This cannot be well handled in traditional Hough voting [17], especially when there is not sufficient amount of training data. We have two steps for the iterative

refinement: 1) fix the scale, search for the activity center with Hough voting; 2) fix the activity center, and determine the scale $\rho$ based on back-projection. We iterate the two steps until convergence.

The initial scale information $\rho$ is set to the average scale of the training videos. Based on the Hough voting step discussed in Section 3, we can obtain the rough position of the activity center. Then back-projection, which has been used in [17][14] for 2D object segmentation or localization, is used to determine the scale parameters.

After the Hough voting step, we obtain a back-projection score for each testing interest point $d_s$ from the testing video based on Eq. 2:

$$\begin{aligned} s_{d_s} &= \textstyle\sum_{d_r \in \mathcal{R}} p(l^*|l_s, l_r) p(\mathbf{f}_s|\mathbf{f}_r) \\ &= \tfrac{1}{Z} \textstyle\sum_{d_r \in \mathcal{R}} e^{-\frac{||l^* - l^s||^2}{\sigma^2}} p(\mathbf{f}_s|\mathbf{f}_r), \end{aligned} \tag{11}$$

where $l^*$ is the activity center computed from last round; $Z$ and $\sigma^2$ are, respectively, normalization constant and kernel bandwidth, which are the same as in Eq. 5. $p(\mathbf{f}_s|\mathbf{f}_r)$ is computed by Eq. 6. The back-projection score $s_{d_s}$ represents how much this interest point $d_s$ contributes to the voting center, i.e., $l^*$. For each sub-volume detected in previous Hough voting step, we first enlarge the original sub-volume in both spatial and temporal domains by 10%. We refer to the extended volume as $V^{l^*}_{W \times H \times T}$, meaning a volume centered at $l^*$ with width $W$, height $H$ and duration $T$. We need to find a sub-volume $V^{l^*}_{w^* \times h^* \times t^*}$ to maximize the following function:

$$\max_{w^*, h^*, t^*} \sum_{d_s \in V^{l^*}_{w^* \times h^* \times t^*}} s_{d_s} + \tau w^* h^* t^*, \tag{12}$$

where $\tau$ is a small negative value to constrain the size of the volume.

We assume each interest point which belongs to the detected activity would contribute in the Hough voting step, i.e., it should have a high back-projection score $s_{d_s}$. Thus, for those interest points with low back-projection scores, we consider them as the background. This motivates us to use the method in Eq. 12 to locate the optimal sub-volume $V^{l^*}_{w^* \times h^* \times t^*}$.

Once we obtain the scale information of the sub-volume, we replace $\rho$ in Eq. 1 with $[w^*, h^*, t^*]$ computed from Eq. 12 and start a new round of Hough voting. The process iterates until convergence or reaching to a pre-defined iteration number.

For activity classification, since the testing videos have already been segmented, the scale $\rho$ can be determined by the width, height and duration of the testing video. The similarity between the training activity model and testing video defined in Eq. 1 is $S(V(\mathbf{x}^*, t^*, \rho), \mathcal{R})$ where $[\mathbf{x}^*, t^*]$ refers to the center position of the testing video.

## 6 Experiments

Two datasets are used to validate the performance of our algorithms. They are UT-Interaction [10] and TV Human Interaction dataset [19]. We perform two

types of tests: 1) activity recognition with few training examples but we have a large testing data (building RPT on the testing data), and 2) activity recognition when the training data is sufficient (building RPT on the training data).

### 6.1 RPT on the Testing Data

In the following experiments, we first show that our algorithm is able to handle the cases when the training data is not sufficient. Experiments on UT-Interaction dataset and TV Human Interaction validate the performance of our algorithm. In these experiments, we build RPT using the testing data without labels. The reason why we do not train our RPT with both the training and testing data is that we need to handle the activity search problem, where we do not have the prior knowledge on query (training) data initially.

**Table 1.** Comparison of classification results on UT-Interaction (20% training)

| Method | [10] | [12] | NN + HV | RPT + HV |
|---|---|---|---|---|
| Accuracy | 0.708 | 0.789 | 0.75 | **0.854** |

**Activity Classification on UT-Interaction Dataset with 20% Data for Training.** We use the setting with 20% data for training and the other 80% for testing on UT-interaction dataset. This evaluation method has been used in [10][12]. Since the training data is not sufficient, we build our random projection trees from the testing data. We list our results in Table 1. "NN + HV" refers to the method that nearest neighbor search is used to replace RPT for feature points matching. It shows that our algorithm has significant performance advantages compared with the state-of-the-arts.

**Activity Classification on UT-Interaction Dataset with One Video Clip only For Training.** [5] provided the result of activity classification with training on only one video clip for each activity type and testing on the other video clips. To compare with [5], we performed the same experiments with just a single video clip as the training data for each activity type. We obtain an average accuracy of 73% which is significantly better than the average accuracy of 65% as reported in [5].

**Activity Search with Localization on UT-Interaction Dataset.** The activity search experiments are tested on the continuous UT-Interaction dataset. In the application scenario of activity search, there is usually just a few or even a single training sample available that indicates what kind of activity the user wants to find. Following the requirement of such an application scenario, we test our algorithm with only one query sample randomly chosen from the segmented videos. But if more training samples are available to our algorithm, the performance will be further boosted. With the help of our iterative activity search

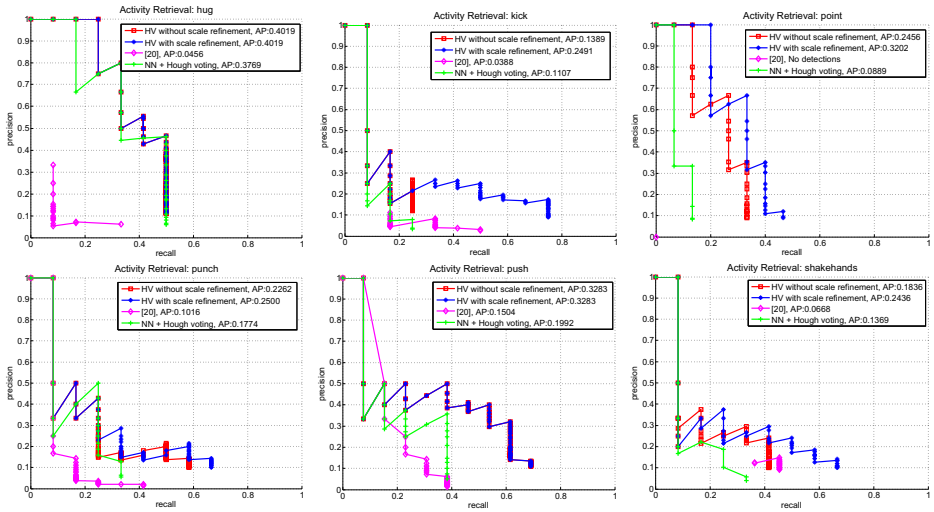


**Fig. 2.** Activity search results on UT-Interaction dataset

algorithm, we can efficiently locate all similar activities in a large un-segmented (continuous) video set. To compute the precision and recall, we consider a correct detection if: $\frac{\text{Volume}(V^* \cap G)}{\text{Volume}(V^* \cup G)} > \frac{1}{2}$ where $G$ is the annotated ground truth subvolume, and $V^*$ is the detected subvolume.

Fig. 2 shows the results of different algorithms. The difference between our activity search and previous work is that we are only given one query video clip. Our system has no prior information about the number of activity categories in the database. In contrast to [10][5], for every activity type, there is at least one video clip provided as training data. As previous works of activity search do not provide precision-recall curves, we only compare with the following algorithms: Branch&Bound [20][2] (magenta curve) and nearest neighbors+Hough voting without scale determination (green curve). We use the same code provided by [20] to run the results. We list two categories of our results: 1) red curves: results after one step of Hough voting without scale refinement and 2) blue curves: results after one round of iteration (including both Hough voting and scale refinement). Compared with NN search, we can see the clear improvements by applying RPT to match feature points. Besides, back-projection refines the results from Hough voting. Since the dataset does not have very large spatial and temporal scale changes, we only present the results after one round of our iterative algorithm. The performance does not improve significantly when we further increase the number of iterations.

Fig. 3 provides sample results of our activity search algorithm. One segmented video (sample frame for each category is shown in the first column) is used as the query and three detected results (marked with red rectangle) are included from the second to the forth column of Fig. 3.

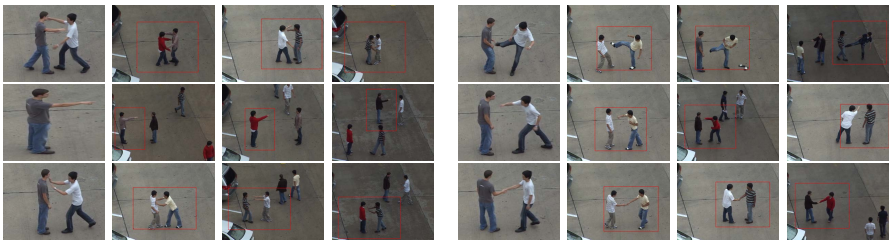

**Fig. 3.** Activity search results on the UT-Interaction Dataset. we show two categories of results in each row. For each category, the first image is from the query video and the following three images are sample detection results. The red regions refer to our detected sub-volumes.

**Activity Search on TV Human Interaction Dataset.** Since UT-Interaction is recorded in controlled environments, we use the TV Human Dataset [19] to show that our algorithm is also capable of handling activity recognition in uncontrolled environments. The dataset contains 300 video clips which are segmented from different TV shows. There are four activities: *hand shake, high five, hug* and *kiss*.

We have performed an experiment on the TV Human dataset for the performance evaluations on different number of training samples. We take the experiment with the following setting: 100 videos (25 videos for each category) as the database and randomly select a number of other videos as queries. RPT is built on the database (testing data). Fig 4 (Left) compares our results with those of NN + Hough voting. It shows the performance benefits of our RPT based matching compared with nearest neighbor based matching.

### 6.2 RPT on the Training Data

We have two experiments to further show that our algorithm can also have promising results for the traditional activity recognition problem, i.e., the training data is sufficient. One is tested on UT-Interaction dataset with Leave-one-out cross validation and another is on TV Human dataset.

**Activity Classification on UT-Interaction Dataset with Leave-One-Out Validation.** This setting was used in the activity classification contest [8]. It is a 10-fold leave-one-out cross validation. Table 2 lists the published results on two different sets of videos. Since enough training data is provided, we build our unsupervised random projection trees from the training data without using the labels. The experimental results show that our algorithm outperforms the state-of-the-art methods on the classification problem when the amount of training data is sufficient.

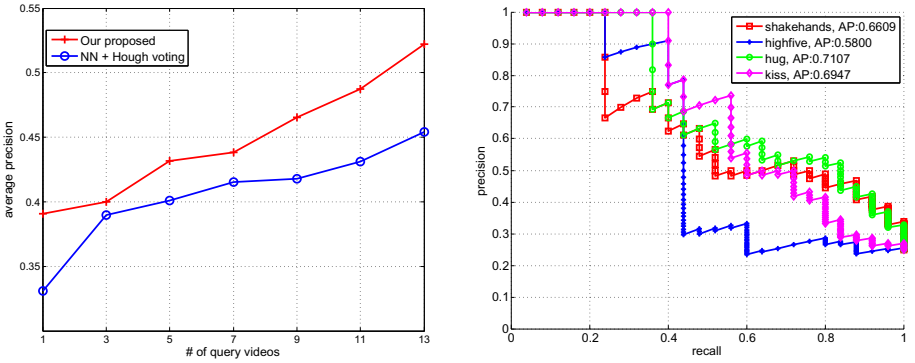


**Fig. 4. Left**: Average precision versus different number of training videos provided on TV Human Dataset (testing on a database with 100 videos). **Right**: PR curves for activity recognition on TV Human Dataset (25 videos for each activity used for training).

**Table 2.** Comparison of classification results on UT-Interaction **Set 1** (left) and **Set 2** (right) with leave-one-out cross validation setting

| Method | Shake | Hug | Kick | Point | Punch | Push | Total |
|---|---|---|---|---|---|---|---|
| [1] + kNN | 0.18 | 0.49 | 0.57 | 0.88 | 0.73 | 0.57 | 0.57 |
| [1] + Bayes | 0.38 | 0.72 | 0.47 | 0.9 | 0.5 | 0.52 | 0.582 |
| [1] + SVM | 0.5 | 0.8 | 0.7 | 0.8 | 0.6 | 0.7 | 0.683 |
| [13] + kNN | 0.56 | 0.85 | 0.33 | 0.93 | 0.39 | 0.72 | 0.63 |
| [13] + Bayes | 0.49 | 0.86 | 0.72 | 0.96 | 0.44 | 0.53 | 0.667 |
| [13] + SVM | 0.8 | 0.9 | 0.9 | 1 | 0.7 | 0.8 | 0.85 |
| [7] | 0.7 | 1 | 1 | 1 | 0.7 | 0.9 | 0.88 |
| [6] BoW | - | - | - | - | - | - | 0.85 |
| Our proposed | 1 | 1 | 1 | 1 | 0.6 | 1 | **0.933** |

| Method | Shake | Hug | Kick | Point | Punch | Push | Total |
|---|---|---|---|---|---|---|---|
| [1] + kNN | 0.3 | 0.38 | 0.76 | 0.98 | 0.34 | 0.22 | 0.497 |
| [1] + Bayes | 0.36 | 0.67 | 0.62 | 0.9 | 0.32 | 0.4 | 0.545 |
| [1] + SVM | 0.5 | 0.7 | 0.8 | 0.9 | 0.5 | 0.5 | 0.65 |
| [13] + kNN | 0.65 | 0.75 | 0.57 | 0.9 | 0.58 | 0.25 | 0.617 |
| [13] + Bayes | 0.26 | 0.68 | 0.72 | 0.94 | 0.28 | 0.33 | 0.535 |
| [13] + SVM | 0.8 | 0.8 | 0.6 | 0.9 | 0.7 | 0.4 | 0.7 |
| [7] | 0.5 | 0.9 | 1 | 1 | 0.8 | 0.4 | 0.77 |
| Our Proposed | 0.7 | 0.9 | 1 | 0.9 | 1 | 1 | **0.917** |

As we can see from Table 2, results from cuboid features [13] are better than those from STIP features [1]. Even though we use STIP features, we still achieve better results than the state-of-the-art techniques that use cuboid features.

**Action Recognition on TV Human Dataset.** We test our algorithm using the standard setting as in [19]: training with 25 videos for each activity and testing on the remaining videos. In addition to [19], there are other works that published the results on this dataset. But they used additional information provided in the dataset, e.g., actor position, head orientation and interaction label of each person. Thus, it is un-fair for us to compare with them since we only utilize the video data.

Following the evaluation method in [19], we also evaluate our algorithm based on average precision. Table 3 compares our results with those reported in [19]. "+Neg" means we add 100 negative videos that do not contain the target activities into the testing dataset. The precision-recall curves from our algorithm are shown in Fig. 4 (Right).

**Table 3.** Comparison of activity classification on TV Human Dataset based on average precision

| | 100 Videos | 100 Videos + 100 Neg |
|---|---|---|
| [19] | 0.3933 | 0.3276 |
| Our algorithm | **0.6616** | **0.5595** |

### 6.3 Computational Complexity

Here we only discuss the online computational cost as the RPT can be built offline. For the Hough voting step, it takes $O(N_M) + O(W'H'T')$, where $N_M$ refers to the number of matches, which is defined in Section 4.2, and $W', H', T'$ are the width, height and duration of the testing videos, respectively. For the back-projection step, the computational complexity is $O(N_M)+O(WHT)$, where $W, H, T$ are the width, height and duration of the extended sub-volume defined in Section 5 and $T << T'$. It takes approximately 10 seconds to perform the activity classification for each 4-second long testing video and 15 seconds for activity search on a 1 min testing video on the UT-Interaction dataset. The feature extraction takes a few more seconds depending on the length of the video. The system is implemented in C++ and runs on a regular desktop PC.

## 7 Conclusion

Local feature voting plays an essential role in Hough voting-based detection. To enable discriminative Hough-voting with limited training examples, we propose propagative Hough voting for human activity analysis. Instead of matching the local features with the training model directly, by employing random projection trees, our technique leverages the low-dimension manifold structure in the high-dimensional feature space. This provides us significantly better matching accuracy and better activity detection results without increasing the computational cost too much. As the number of trees grows, our propagative Hough voting algorithm can converge to the optimal detection. The superior performances on two benchmarked datasets validate that our method can outperform not only with sufficient training data, e.g., in activity recognition, but also with limited training data, e.g., in activity search with one query example.

**Acknowledgement.** This work was supported in part by the Nanyang Assistant Professorship (SUG M58040015) to Dr. Junsong Yuan.

## References

1. Laptev, I.: On space-time interest points. International Journal of Computer Vision 64(2-3), 107–123 (2005)
2. Yuan, J., Liu, Z., Wu, Y.: Discriminative Video Pattern Search for Efficient Action Detection. IEEE Trans. on PAMI (2011)

3. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: Proc. CVPR (2008)
4. Yuan, F., Prinet, V., Yuan, J.: Middle-Level Representation for Human Activities Recognition: the Role of Spatio-temporal Relationships. In: ECCV Workshop on Human Motion (2010)
5. Gaur, U., Zhu, Y., Song, B., Roy-Chowdhury, A.: A String of Feature Graphs Model for Recognition of Complex Activities in Natural Videos. In: ICCV (2011)
6. Ryoo, M.S.: Human Activity Prediction: Early Recognition of Ongoing Activities from Streaming Videos. In: ICCV (2011)
7. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. PAMI, 2188–2202 (2011)
8. Ryoo, M.S., Chen, C., Aggarwal, J.: An overview of contest on semantic description of human activities, SDHA (2010)
9. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: International Conference on Very Large Data Bases (VLDB), pp. 518–529 (1999)
10. Ryoo, M.S., Aggarwal, J.K.: Spatio-Temporal Relationship Match: Video Structure Comparison for Recognition of Complex Human Activities. In: ICCV (2009)
11. Amer, M.R., Todorovic, S.: A Chains Model for Localizing Participants of Group Activities in Videos. In: ICCV (2011)
12. Brendel, W., Todorovic, S.: Learning Spatiotemporal Graphs of Human Activities. In: ICCV (2011)
13. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior Recognition via Sparse Spatio-Temporal Features. In: Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (2005)
14. Razavi, N., Gall, J., Van Gool, L.: Backprojection Revisited: Scalable Multi-view Object Detection and Similarity Metrics for Detections. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 620–633. Springer, Heidelberg (2010)
15. Choi, W., Savarese, S.: Learning Context for Collective Activity Recognition. In: CVPR (2011)
16. Yao, B., Fei-Fei, L.: Modeling mutual context of object and human pose in human-object interaction activities. In: CVPR (2010)
17. Leibe, B., Leonardis, A., Schiele, B.: Robust Object Detection with Interleaved Categorization and Segmentation. IJCV 77(1-3), 259–289 (2007)
18. Dasgupta, S., Freund, Y.: Random projection trees and low dimensional manifolds. In: ACM Symposium on Theory of Computing (STOC), pp. 537–546 (2008)
19. Patron-perez, A., Marszalek, M., Zisserman, A., Reid, I.: High Five: Recognising human interactions in TV shows. In: BMVC (2010)
20. Yu, G., Yuan, J., Liu, Z.: Unsupervised Random Forest Indexing for Fast Action Search. In: CVPR (2011)
21. Moosmann, F., Nowak, E., Jurie, F.: Randomized clustering forests for image classification. PAMI 30, 1632–1646 (2008)
22. Klaser, A., Marszalek, M.: A spatio-temporal descriptor based on 3D-gradients. In: BMVC (2008)