Detecting and recognizing human action in videos

Yu, Gang

2014

Yu, G. (2014). Detecting and recognizing human action in videos. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/61739>

<https://doi.org/10.32657/10356/61739>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**DETECTING AND RECOGNIZING HUMAN ACTION IN
VIDEOS**

YU GANG

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING

2014

DETECTING AND RECOGNIZING HUMAN ACTION IN VIDEOS

YU GANG

YU GANG

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2014

Acknowledgements

First of all, I want to express my most sincerely gratitude to my supervisor, Professor Junsong Yuan, for his support, encouragement and instruction in the past four years. Professor Junsong Yuan directed me into such a beautiful and fantastic research realm of computer vision and multimedia, and supported me with well-equipped environment and active research atmosphere. Besides, Dr. Zicheng Liu from Microsoft Research Redmond is a wonderful teacher for me, I thank him for his direction, valuable feedbacks and warm encouragement on my research progress. I also want to thank all the members in our research group, Yuan's Group: Yang Cong, Chaoqun Weng, Chunluan Zhou, Du Tran, Gangqiang Zhao, Hui Liang, Hongxing Wang, Jingjing Meng, Jianfeng Ren, Ye Luo, Yuning Jiang, Zhou Ren, Kang Dang, and Yang Xiao. Because of the energetic research style we formed, I was passionate and happy in the last four years. Finally, my gratitude goes to my dear family and all my friends in my life!

Contents

Acknowledgements	i
Summary	vii
List of Figures	x
List of Tables	xii
List of Abbreviations	xiii
1 Introduction	1
1.1 Motivation and Objective	2
1.2 Major Contributions of this Thesis	2
1.3 Organization of this Thesis	5
2 Literature Review	7
2.1 Video Feature and Action Representation	7
2.1.1 Low level Video Feature	8
2.1.2 Middle level Action Representation	9
2.2 Related Work on Human Action Recognition	10
2.3 Related Work on Human Action Detection	11
2.4 Related Work on Human Action Search	14
2.5 Related Work on Human Action Prediction	15
2.6 Related Work on Random Trees	15
2.7 Datasets for Action Recognition and Detection	16

3	Fast Action Detection in Unconstrained Videos	19
3.1	Introduction	19
3.2	Mutual Information based Formulation	23
3.3	Random Forest based Voting	25
3.4	Action Detection and Localization	29
3.4.1	Spatial Down-Sampling	31
3.4.2	Top-K Search and λ Search	34
3.5	Experiments	36
3.5.1	Action Classification	36
3.5.2	Action Detection	38
3.5.3	Computational Cost	42
3.6	Concluding Remarks	45
4	Action Detection with Limited Training Data	46
4.1	Introduction	46
4.2	Background on Hough Voting	49
4.3	Propagative Hough Voting	50
4.3.1	Problem Formulation	50
4.3.2	Interest Point Matching	53
4.3.3	Scale Determination	57
4.4	Applications	58
4.4.1	Human Activity Detection	58
4.4.2	Human Activity Recognition	59
4.4.3	Human Activity Prediction	59
4.5	Experiments	60
4.5.1	Experiments on Human Activity Detection	61
4.5.2	Experiments on Human Activity Recognition	63
4.5.3	Experiments on Human Activity Prediction	67
4.5.4	Computational Complexity	69
4.6	Concluding Remarks	70
5	Interactive Action Detection/Search using Randomized Visual Vo-	

cabularies	71
5.1 Introduction	71
5.2 Randomized Visual Vocabularies for Video Representation	74
5.3 Local Interest Point Matching	77
5.4 Efficient Action Search	81
5.4.1 Hierarchical Sub-volume Search	81
5.4.2 Refinement with Hough Voting	83
5.4.3 Interactive Search	84
5.4.4 Computational Complexity	85
5.5 Experimental results	86
5.5.1 Action Detection	88
5.5.2 Action Retrieval on MSR II	90
5.5.3 Action Retrieval on CMU database	92
5.5.4 Action Retrieval on Youtube Video	93
5.5.5 Action Retrieval on Large-scale Database	95
5.5.6 Implementations	97
5.5.7 Computational Cost	99
5.6 Concluding Remarks	102
6 Conclusions and Future Work	104
6.1 Conclusions	104
6.2 Future Work	106
Bibliography	112

Summary

Detecting and recognizing human actions is of great importance to video analytics due to its numerous applications in video surveillance and human computer interaction. Despite much previous work, fast and reliable action detection and recognition in unconstrained videos remain a challenging problem. First of all, actions are spatio-temporal patterns characterized by both motion and appearance features. The same type of action may exhibit large variations due to the changes of motion speed, scale, view point, clothing, not to mention partial occlusions. It is thus a challenge to perform robust action matching that is insensitive to such variations, especially if only a limited number of training examples are provided. Moreover, fast action detection and localization is another challenging issue in cluttered and dynamic environment. Compared with image based object detection which only requires spatial localization, action localization is in spatio-temporal video space thus is much more time consuming.

This thesis presents a systematic study on detecting and recognizing human actions in cluttered and dynamic environments. The videos are characterized by spatio-temporal local features, and the proposed methods leverage the fast matching of local features to perform action recognition and detection. To capture the intra-class variations of action categories, randomized trees are developed to capture the local feature distribution of the action categories. Such a tree-based indexing enables fast local feature matching, and when limited training examples are available, it can be easily extended to index both labelled and unlabelled data samples and perform semi-supervised learning to improve the detection performance. Even with only one exemplar query action, the randomized tree indexing approach can still achieve promising result to detect similar actions in the big video corpus efficiently. To perform fast spatio-temporal

action localization, two different approaches have been proposed: (1) Coarse-to-fine branch-and-bound search and (2) Propagative Hough voting. Both methods can significantly reduce the computational cost of action localization, and do not rely on human detection, tracking, and background subtraction.

By addressing the fundamental challenges of action detection and recognition, this thesis also investigated action detection solutions for different application scenarios, such as multi-class action detection, action search with one query example, and on-line action prediction based on partial video observation. Extensive experiments on benchmarked datasets show that the proposed methods can achieve promising results compared with the state of the arts.

List of Figures

1.1	Comparison of the tasks of action recognition, action detection, action search, and action prediction. The detail explanation will be presented in Chapter 2.	4
1.2	An overview of this thesis.	5
3.1	An illustration of sub-volume search.	20
3.2	The overview of the proposed random forest based action detection algorithm.	23
3.3	Approximation of the spatial down-sampling. Left figure shows the score image in the original resolution and right figure shows the down-sampled score image.	32
3.4	Precision-recall curves for action detections with different methods. . .	41
3.5	Comparisons of Average Precision-Recall curves.	42
3.6	Detection results (Random Forest+Top-K) of handclapping (1st row), handwaving (2ed row), boxing (3rd row) and walking (4th row) are listed.	43
4.1	An illustration of Propagative Hough Voting.	48
4.2	Activity detection results on UT-Interaction dataset.	60

4.3	Activity detection results on the UT-Interaction Dataset. For each category, the first image is from the query video and the following three images are sample detection results. The red bounding boxes enclose the detected sub-volumes.	62
4.4	PR curves for activity recognition on TV Human Dataset (25 videos for each activity used for training).	66
4.5	Asymptotic experiment on UT-Interaction (20% training).	67
4.6	Human activity prediction on UT-Interaction dataset. (Set 1: Left; Set 2: Right)	68
5.1	Overview of the proposed algorithm.	72
5.2	An illustration of the indexing and action search based on randomized-visual-vocabulary.	79
5.3	Illustration for Hough refinement.	84
5.4	An illustration of the retrieval results (without and with Hough refinement). For each row, the first image represents one query frame and followed with seven highest ranked retrieved results. No user feedback is employed here.	86
5.5	An illustration of sample frames used for the experiments. The first row, second row, and third row show sample frames from MSR II dataset, CMU dataset and Youtube dataset, respectively. Sample frames from a 5-hour large dataset are shown in the forth row.	87
5.6	Handwaving detection on MSR II dataset based on precision-recall curve (AP means the average precision).	88
5.7	Action search on MSR II dataset based on precision-recall curve.	89

5.8	Interactive action retrieval on MSR II dataset based on precision-recall curve.	92
5.9	Retrieval results from CMU dataset. For each row, the first image represents the query frame and the following seven images refer to the highest ranked retrieved results.	93
5.10	An action search example with tennis serve action. Both the query and database videos are downloaded from Youtube. The seven frames in the first row represent the frames in the query video while the images from second to sixth rows show the top-5 retrieved samples.	94
5.11	Handwaving action retrieval in large dataset. The first row illustrates the seven frames from query action while the following five rows give the retrieved examples. The positive examples are marked by cyan and the negative example is marked by yellow.	96
5.12	Handclapping action retrieval in large dataset. The first row illustrates the seven frames from query action while the following five rows give the retrieved examples. The positive examples are marked by cyan and the negative example is marked by yellow.	97
5.13	Boxing action retrieval in large dataset. The first row illustrates the seven frames from query action while the following five rows give the retrieved examples. The positive examples are marked by cyan and the negative example is marked by yellow.	98
5.14	Ballet spin action retrieval in large dataset. The first row illustrates the seven frames from query action while the following five rows give the retrieved examples. The positive examples are marked by cyan and the negative example is marked by yellow.	99

5.15	Action retrieval results from 5-hour large dataset.	100
5.16	The evaluation of average precision based on the number of trees. . . .	101

List of Tables

2.1	Action Recognition Datasets.	17
2.2	Action Detection Datasets.	17
3.1	Confusion matrix for KTH action dataset. The total accuracy is 91.8%.	38
3.2	Comparison of different reported results on KTH dataset.	39
3.3	Time consumed for voting one STIP and one video sequence (for example, 10000 STIP points). Only CPU time is considered.	43
3.4	A comparison with the computational cost.	44
3.5	Comparison of total time cost for action detection. Only CPU time is considered.	45
4.1	Comparison of classification results on UT-Interaction Set 1 with leave-one-out cross validation setting.	64
4.2	Comparison of classification results on UT-Interaction Set 2 with leave-one-out cross validation setting.	64
4.3	Comparison of activity classification on TV Human Dataset based on average precision.	65

4.4	Comparison of recognition results on UT-Interaction (20% training). . .	65
5.1	List of Datasets for experiments	87
5.2	CPU time consumed by STIP voting in MSR II dataset which consists of 870,000 STIPs.	102
5.3	Total computation time of the retrieval system.	102

List of Abbreviations

ABBREVIATIONS

BoW

B&B

HoF

HoG

HT

HV

LSH

MAP

MBH

NN

PCA

RF

RPT

STIP

STISM

SVM

FULL EXPRESSIONS

Bag of Word

Branch and Bound

Histogram of Flow

Histogram of Gradient

Hough Transform

Hough Voting

Local Sensitive Hashing

Maximum A Posterior

Motion Boundary Histogram

Nearest Neighbor

Principal Component Analysis

Random Forest

Random Projection Tree

Space-Time Interest Point

Spatial-Temporal Implicit Shape Model

Support Vector Machine

Chapter 1

Introduction

Over the last decade, with the proliferation of surveillance cameras and camera equipped mobile devices, there has been an exponential growth in the number of videos being captured and archived. Detecting and recognizing human actions are of great importance for analyzing these videos due to various applications. For example, in video surveillance, it is important to detect specific human behaviours such as picking up something from the floor or making a phone call in the crowd. For healthcare, detecting a falling down activity of a senior people at home can enable timely rescue. For unmanned aerial vehicle, detecting specific activity in the battle field, such as burying a mine, is of great importance too. For egocentric vision, understanding the human behaviour plays an important role to sense the environment. For all these applications, detecting and understanding human behaviors are the core problems that need to be solved. Despite much previous work, efficient and robust action detection still remains a challenging problem.

1.1 Motivation and Objective

Overall, there are two major challenges for human action detection and recognition. First of all, actions are spatio-temporal patterns characterized by both motion and appearance features. The same type of action may exhibit large variations due to the changes of motion speed, scale, view point, clothing, not to mention partial occlusions. It is thus a challenge to perform robust action matching that is insensitive to such variations. Moreover, if insufficient training examples are available, how to model the large intra-class variation will become more challenging.

Second, efficient action localization in unconstrained videos is a non-trivial issue but is critically required. In many applications, real-time detection and recognition of human actions is highly demanded. Compared with image based object detection which only requires spatial localization, action detection and localization in spatio-temporal video space thus is much more time consuming. For video surveillance in dynamic and crowded scenes, we cannot rely on background subtraction to quickly identify the humans in the foreground and then recognize their actions. Extra efforts are thus required. For moving cameras, such as in the egocentric vision applications, they make the problem even worse as we need to search for the whole screen to find the target action. Efficient action analysis is in general a quite challenging problem in such unconstrained videos.

1.2 Major Contributions of this Thesis

To address these two fundamental challenges in human action detection and recognition problems, a systematic study has been conducted in this thesis.

- (1) For the challenge of large intra-class variation, multiple visual vocabularies are

utilized. Instead of generating or learning one set of vocabulary to represent the human actions, multiple sets of vocabularies can essentially relieve the problem of large intra-class variation. To implement the vocabularies, different tree variations have been developed. Supervised trees in Chapter 3 optimizes the classification error to obtain better recognition accuracy. Unsupervised random projection tree in Chapter 4 improves the matching accuracy by leveraging the underlying data distribution from both labeled and unlabeled video data while the goal of random indexing tree in Chapter 5 is to provide efficient local interest point matching.

(2) For the challenge of efficient action localization, two practical solutions are proposed for different scenarios. Propagative Hough voting in Chapter 4 optimizes both the localization performance as well as the computational cost. Hough voting is applied to the local interest points and a small set of candidate sub-volumes are generated. Since the spatio-temporal configuration of the local interest points has been utilized during the voting step, the localization accuracy can be largely improved. However, despite the superior performance of propagative Hough voting, it is still not faster enough for action localization in a large-scale dataset. Coarse-to-fine branch-and-bound search in Chapter 5, on the other hand, employs a sliding-window approach to enumerate all the potential sub-volumes in the video space. Due to the extremely large number of sub-volumes, speed-up techniques like Top-K search and spatial-downsampling are developed in Chapter 3 to cut the search space and reduce the computational cost. In addition, a coarse-to-fine strategy is applied to the search process which can significantly improve the computational speed. Although coarse-to-fine branch and bound search can obtain significant speed improvement, there is a performance compromise since the spatio-temporal configuration information among the local interest points has been discarded to speed up the matching.

To validate the performance of the proposed algorithms, four categories of human

action understanding tasks are considered. The four tasks are: human action recognition, action detection, action search, and action prediction. Human action recognition is to determine the category of the segmented video sequence based on a set of labeled training data. Action detection, however, not only needs to specify the category of the action but also spatially and temporally locate the action from an unsegmented video sequence. Different from action detection and recognition, where a set of videos including both the positive and negative video sequences are available for training a recognition system, the training data for action search is often based on one query sequence. Given the query video, the goal of action search is to find the semantically similar actions from an unlabeled video database. Action prediction is trying to predict the category of action sequence based on the partial observation. This is of great significance for many online applications. A comparison of these four tasks is illustrated in Fig. 1.1. A literature review can be found in Chapter 2.

	Action Recognition	Action Detection	Action Search	Action Prediction
Test Video X
Description	1. Segmented multi-class training data 2. Segmented test data	1. Segmented multi-class training data 2. unsegmented test data	1. One query segmented video 2. unsegmented database	1. Segmented multi-class training data 2. Segmented partial observation
Goal	$f(X) \rightarrow \{1, \dots, K\}$	$f(X) \rightarrow [x, w, y, h, t, d]$	$f(q) \rightarrow [x, w, y, h, t, d, i]$	$f(O) \rightarrow \{0, 1, \dots, K\}$ Where $O \subset X$

Figure 1.1: Comparison of the tasks of action recognition, action detection, action search, and action prediction. The detail explanation will be presented in Chapter 2.

Figure 1.2: An overview of this thesis.

1.3 Organization of this Thesis

Fig. 1.2 gives an outline of this thesis. Chapter 1 gives an introduction to the thesis and presents the problems to address. Chapter 2 provides a literature survey on the recent developments in human action detection and recognition. Then three different application scenarios are discussed: action detection with sufficient training data in Chapter 3, action detection with limited training data in Chapter 4, and action search (one extreme case of action detection) with one query sample in Chapter 5.

More specifically, in Chapter 3, a novel supervised algorithm is developed for efficient spatio-temporal localization of human actions in video sequences. The algorithm improves upon the state of the art on two aspects. First, a random forest based voting technique is proposed to compute the scores of the interest points, which achieves multiple orders-of-magnitude speed-up compared to the nearest neighbor based scoring scheme. Second, a top-K search technique is presented which can detect multiple action instances simultaneously with a single round of branch-and-bound search. To

reduce the computational complexity of searching higher resolution videos, subvolume search is performed on the down-sampled score volumes.

In Chapter 4, Propagative Hough voting, which leverages both the labeled and unlabeled video data, is proposed to improve local interest point matching. More specifically, by performing feature matching with random projection trees, this algorithm can leverage the underlying data distribution and obtain promising results even when the training data is limited. Moreover, since the propagative Hough voting formulation is additive to the test interest points, the algorithm can be easily extended for human action prediction problem. Furthermore, with a scale refinement step, this algorithm can determine the spatio-temporal location of human actions when the searched videos are not segmented.

In Chapter 5, an unsupervised action search algorithm is proposed which can effectively locate the subvolumes similar to the query video. Random-indexing-trees based visual vocabularies are introduced and demonstrated to perform well for the database indexing. By increasing the number of vocabularies, the large intra-class variation problem can be relieved despite only one query sample available. Moreover, superior action search performance can be obtained with the help of the interactive feedback. Last but not least, a coarse-to-fine subvolume search scheme is proposed, which results in a dramatic speedup over the existing branch-and-bound method.

Finally, Chapter 6 will conclude this thesis and propose three potential future research directions.

Chapter 2

Literature Review

As discussed in Chapter 1, there are four categories of human action tasks considered in this thesis. The fundamental part shared by those four tasks is the video representation. Thus, in this chapter, traditional action feature and video representation will be first discussed. Later, literature reviews on the four action understanding tasks will be covered separately.

2.1 Video Feature and Action Representation

Generally speaking, low-level video feature and middle-level video feature are two popular ways to encode the video content. Low-level video feature is usually extracted from the local interest point or densely sampled from the videos without the physical meaning. Middle-level video feature, on the other hand, encodes the video content by middle-level components like attributes and templates.

2.1.1 Low level Video Feature

Similar to image feature, low-level video feature extraction usually contains two components: local interest point detector and descriptor. For the interest point detector, there are several mature approaches. Harris 3D detector is employed in Spatial-Temporal Interest Point (STIP) [1] and the spatial and temporal scale of the interest point are determined by iteratively maximizing the normalized Laplace operator. Since the extracted interest point is very sparse, it saves a lot of computational and memory cost based on the STIP feature. Hessian 3D detector is presented in [78] with more dense interest point sampling. Box filter and integral video are employed in [78] to reduce the computational cost. Cuboid detector [79] tries to detect local interest point by the response of Gabor filter. Recently, dense trajectory has been proposed in [80] with promising results on various benchmarked action recognition datasets. Empirically, it is difficult to claim which local interest point is superior to the others. Usually, the selection of local interest point depends on the specific application scenario. For instance, if computational cost is a major concern, STIP would be a good choice compared with dense local features. On the other hand, if computational issue is not a problem, dense trajectory should be better due to the promising performance for action recognition.

Given the extracted interest point, a 3D patch near the interest point is extracted to describe the local point. Histogram of Gradient (HoG) and Histogram of Optical Flow (HoF) [1] are widely used to represent the local interest point. HoG encodes the gradient information while HoF encodes the motion information of the local interest point. In addition to HoG and HoF, HoG3D [81] and Motion Boundary Histogram (MBH) [145, 80] are useful interest point descriptor. HoG3D [81] is based on histograms of oriented 3D spatio-temporal gradients. A memory-efficient algorithm based on integral videos is presented in [81] to reduce the feature extraction cost. MBH is first presented in [145] for human detection, which encodes the relative motion between

the pixels. Similar to HoG3D [81], a 3-dimension SIFT descriptor is presented in [128] to encode the local interest point. An overview of local interest point can be found in [82].

In this thesis, low-level video feature is employed for video representation. STIP is adopted as the local interest point detector and HoG/HoF is utilized as the descriptor for the local interest points. For all the algorithms covered in the thesis, all interest points from the whole video will be processed. When performing action recognition and detection, interest points are matched based on HoG and HoF representation. Only those interest points with similar motion and appearance to the training interest points (from human actions) will be given positive votes.

2.1.2 Middle level Action Representation

Different from low-level video feature, middle-level feature representation is usually constructed based on the low-level features. Template based representation is one popular middle-level representation. In [46], space-time locally adaptive regression kernels and the matrix cosine similarity measure are developed for action recognition with a single example. A novel compact local descriptor of video dynamics is presented in [26] and an associated similarity measure is introduced which enables efficient exhaustive search. Similarly, in [16], it presents a template-based method called Action MACH, which is based on a Maximum Average Correlation Height (MACH) filter. The idea under MACH is to synthesize a filter for a given action class so as to capture intra-class variability.

In addition, there are a number of interesting works without specific template modeling. In [83], attributes are employed for action recognition, which can be either manually labeled or learnt from the data. In [84], “action bank” is proposed by combining

a set of individual detectors. Similar ideas have been proposed in [89, 91, 92]. The method in [89] presents a data-driven approach to learn motionlets, which can be considered as a cluster based on motion and appearance, from the training videos. Salient spatio-temporal structure based on the clusters of trajectories is employed in [91] as a middle-level representation for action classification. The method in [92] can automatically mine the informative spatio-temporal patch in the video for action recognition. Compare with template based approach, these approaches are usually more flexible to encode the discriminative information for video representation.

Based on the action feature and representation, various human action applications can be developed. In the following sections, a brief discussion of recent work on action recognition, action detection, action search, and action prediction will be presented, respectively. The term “action” and “activity” will be exchangeable in the thesis. Also, “action recognition” and “action classification” are exchangeable.

2.2 Related Work on Human Action Recognition

As shown in Fig. 1.1, action recognition is to perform action classification based on segmented video sequences. The topic of human action recognition has been widely exploited in the recent decade. Bag-of-word (BoW) representation followed by a SVM classifier based on the low-level video representation as in Section 2.1.1 is a standard approach for human action recognition. Different low-level features have been employed in this setting, such as STIP [1][3], Cuboid feature [79], and dense trajectory [80][95]. Similarly, BoW representation is also discussed in [4][72]. In [4], it utilizes the movie script to annotate the video sequences and a multi-channel SVM is employed for the human action classification. In [72], probabilistic latent semantic analysis model is used to learn the probability distributions of words and topics corresponding to the human

actions. Apart from BoW representations, Gaussian Mixture Models (GMM) [14] and nearest neighbor (NN) search [9][96] are alternative solutions to action classification.

Graphical model is another widely used approach for action recognition. In [97], conditional random fields as well as hidden Markov model are evaluated for action recognition. An extended probabilistic latent semantic analysis is presented in [98], which captures both the semantic and structure information for action recognition. A hierarchical Bayesian model which connects low-level features, atomic action and multi-agent interaction is proposed in [99]. In [100], time delayed probabilistic graphical model is used to model the activity global temporal dependencies. A novel string representation is proposed in [101], which encodes the spatio-temporal ordering of local interest points. Spatio-temporal graphs are constructed to model the relationship between the multiscale video segments in [102].

Besides from BoW and graphical model based algorithms, other interesting works for action recognition include [103, 107, 75]. In [103], slow feature analysis based approach is presented which is able to extract useful motion patterns and improve the classification results. A subspace forest is proposed to perform the set-to-set feature matching in [104]. In [105], action recognition is motivated by the success of speech recognition and a novel scheme adapted from speech recognition is presented with promising results on action recognition. Context information is integrated in [107] and [75] for action recognition. A deep learning based spatio-temporal feature is proposed in [106] which shows good results for many action recognition datasets.

2.3 Related Work on Human Action Detection

Different from action recognition, action detection not only needs to determine the action category but also spatially and temporally locate the sub-volume of the actions.

Thus, it is usually more challenging than action recognition. Depending on how to determine the action location, the existing approaches for human action detection can be roughly classified as three categories: sliding window-based approach, voting based approach, and tracking-based approach.

The sliding window-based approach tries to enumerate all the sub-volumes in the spatio-temporal video space. Usually a template is constructed to model the spatio-temporal information for each action category [13, 10, 16, 26, 40]. Two types of temporal templates are proposed in [13] for characterizing actions: (1) the motion energy image (MEI), which is a binary image that encodes locations where motion has occurred; and (2) the motion history image (MHI), which is scalar valued image whose intensity is a function of the recent motion. In [18], the motion history volume (MHV) is introduced as a free-viewpoint representation for human actions. To better handle the cluttered and dynamic backgrounds, an input video is over-segmented into many spatio-temporal video volumes in [11]. An action template is matched by searching among these over-segmented video volumes. In [46], space-time local regression kernels are densely computed and the cosine similarity is employed to measure the distance between the query video and target video. However, because only one template is utilized, previous template-based methods usually have difficulties in handling intra-class action variations. Some discriminative methods have been developed to improve template matching. In [19, 21], Haar features are extended to 3-dimensional space, and boosting is applied to integrate these features for final classification. In [23], a successive convex matching scheme is proposed for action detection. In [22], a prototype-based approach is introduced, where each action is treated as a sequence of prototypes. However, the computational cost of these algorithms is extremely high. For example, it takes several minutes or even hours to handle a single short video clip. More specifically, due to the need to enumerate all the possible subvolumes in a video clip, the computational

complexity grows rapidly as the templates become more complex. In Chapter 3, a revised branch-and-bound based search algorithm will be proposed which significantly reduces the computational cost of sliding window-based approach. The computational cost will be further reduced in Chapter 5 with a novel coarse-to-fine framework.

Different from sliding window-based approach, voting-based approach generates a set of potential sub-volumes for evaluation. Without enumerating all the 3D sub-volumes, voting based approach can naturally reduce the computational cost. Frame-level Hough voting is utilized in [94] to generate the potential human locations and a max sub-path search algorithm is employed to locate the human actions in the videos. The method in [129] presents a unified framework for human detection and action recognition based on Hough voting. Since the human location is first detected in the videos, the main goal of [129] is to perform temporal localization. In Chapter 4, propagative Hough voting will be proposed on the spatio-temporal domain followed by a scale refinement algorithm to determine the sub-volumes of human actions.

The third category for human action detection is the tracking based approach. Pedestrian detection and tracking algorithms are used to determine the human location for each video frame. In [12], multiple instance learning based approach is applied to human action detection, which relies on head detection and tracking. Similarly, the techniques presented in [42] and [43] also require human tracking as a pre-processing step. Tracking based approach is largely constrained by the tracking precision. Since human tracking in a complex and dynamic environment is itself a challenging problem, it is not practical to rely on tracking to solve the action detection problem.

2.4 Related Work on Human Action Search

Despite a lot of works in action recognition and detection reported, action retrieval, on the other hand, is less exploited. Most of the existing action retrieval algorithms can be roughly divided into two classes based on the number of query samples. Algorithms in the first category [26, 64] perform the sliding window search on the database with a single query sample. The idea for both [26] and [64] are representing query and database videos with the same features and comparing the similarity based on query-to-subvolume measurement. In [26], visual space-time oriented energy measurement is used while a five-layer hierarchical space-time model is employed in [64]. One limitation of these techniques is that, with a single query sample, it is impossible to model the action variations. Besides, an action retrieval system usually involves user interactions but their approaches do not have the capability to incrementally refine their models based on the user feedback. The other category of action retrieval algorithms, for example [71], is based on a set of query samples, usually including both positive and negative samples. Despite the fact that they work well in uncontrolled videos, the computational cost is high and they would fail if insufficient number of query samples are provided. Apart from the above work, there exist some other interesting algorithms in the literature. For example, the methods in [65, 66, 67] rely on auxiliary tools like storyboard sketches, semantic words and movie transcripts for action retrieval, while the method in [69] is specifically focused on quasi-periodic events. In [68], action retrieval based on static images is reported while the algorithm in [70] retrieves the similar human action patterns with spatiotemporal vocabulary.

Although interesting works have been presented on the problem of human action search, there are still two challenges to be addressed: computational cost and interactive search. On one hand, the response time for action search should be fast otherwise

the user’s experience would be poor. On the other hand, as one round of action search may not provide the demanded performance, user feedback is highly demanded to improve the search accuracy. To address these two challenges, Chapter 5 will present an efficient and interactive action search algorithm.

2.5 Related Work on Human Action Prediction

The problem of human action prediction has been proposed in [109]: *inference of the ongoing action given temporally incomplete observations*. Integral bag-of-words (BoW) and dynamic bag-of-words are proposed in [109] to enable action prediction with only partial observations. Despite certain successes of [109], it still has several limitations. First, since the BoW model ignores the spatial-temporal relationships among interest points, it is not discriminative enough to describe the human actions. Also, although integral BoW and dynamic BoW in [109] consider the temporal information by matching between sub-intervals, there lacks a principled way to determine the optimal interval length. In [108], a max-margin structure SVM is proposed to early detect events as well actions.

In Chapter 4, an additive Hough voting formulation will be proposed and superior performance has been obtained for the challenging action prediction problem on segmented test videos.

2.6 Related Work on Random Trees

Since tree structure serves an important role for local interest point matching in this thesis, a brief review of recent development of random trees is necessary. Originally, tree structure is used for nearest neighbor search. KD-tree [111, 112] tries to index the

data points by hierarchically splitting on different feature dimensions. A priority queue is utilized to reduce the search cost and the exact nearest neighbor can be obtained by a back-trace step. Approximated nearest neighbor search can be efficiently computed as well with multiple KD-trees. Similar idea is utilized in VP-tree [112], random projection tree [114] and PCA tree [113]. Unsupervised random trees are also employed in [41, 6] for action recognition and detection.

Supervised random trees, on the other hand, have been widely used as a classifier for many computer vision and multimedia applications [117, 116, 115, 25, 51]. The central idea is to construct the tree based on the supervised label information. Besides, semi-supervised random trees like [118, 85] make a good balance between supervised classification and unsupervised data distribution.

In Chapter 3, supervised random forest will be adopted for local interest point classification. Also, unsupervised random trees will be employed in Chapter 4 and Chapter 5 to improve the local interest point matching. Random projection tree in Chapter 4 tries to improve the matching accuracy by leveraging the underlying data distribution from both labeled and unlabeled video data while random indexing tree in Chapter 5 is to provide efficient local interest point matching.

2.7 Datasets for Action Recognition and Detection

In this section, the widely used datasets for action recognition and detection will be reviewed. Let us start from datasets for action recognition. Table 2.1 lists popular action recognition datasets. KTH [3], Weizmann [77], and IXMAS [119] are recorded in constrained environments while the other datasets are collected either from Youtube videos or movies. Besides, IXMAS [119] is a cross-view dataset which can be employed for action recognition with multiple views.

2.7. Datasets for Action Recognition and Detection

Dataset	No. of classes	No. of videos	camera motion	Release Year
KTH [3]	6	600	slight	2004
Weizmann [77]	9	81	No	2005
IXMAS [119]	11	165	No	2006
UCF Sports [16]	9	182	Yes	2009
UCF11 [35]	11	1168	Yes	2009
UCF50 [120]	50	6681	Yes	2010
UCF101 [121]	101	13320	Yes	2012
Hollywood I [4]	8	430	Yes	2008
Hollywood II [75]	12	3669	Yes	2009
TV Human [122]	4	300	Yes	2010
Olympic [39]	16	800	Yes	2010
HMDB51 [123]	51	6766	Yes	2011

Table 2.1: Action Recognition Datasets.

Although there are many action recognition datasets available, the number of action detection datasets is much smaller compared with action recognition datasets. Some of the widely used action detection datasets are listed in Table 2.2. MSRII dataset [9] and UT-Interaction dataset [109] are recorded in constrained environments while VIRAT [73], UCF [121], and MPII Cooking [124] datasets are collected from unconstrained conditions. More specifically, VIRAT [73] dataset is targeted at surveillance videos; UCF [121] dataset is trying to solve the detection problem in Youtube clips; MPII Cooking [124] dataset is for action recognition and detection in the kitchen environment. Similar datasets for action recognition in the kitchen include [125][126].

Dataset	No. of classes	ave No. of instances	Resolution	Release Year
MSRII [9]	3	68	320×240	2009
UT-Interaction [109]	6	20	720×480	2010
VIRAT [73]	23	10-1500	1920×1080	2011
UCF [121]	24	100	320×240	2012
MPII Cooking [124]	65	86	1624×1224	2012

Table 2.2: Action Detection Datasets.

No dataset has been presented specifically for action search or action prediction. But the datasets discussed in Table 2.1 and Table 2.2 can be used to evaluate the algorithms for action search and prediction.

In this thesis, KTH dataset will be employed in Chapter 3 for action recognition experiments. In Chapter 3 and Chapter 5, KTH has been used as the training (or query) set for action detection (or search). For the test dataset of action detection, MSRII dataset is employed in Chapter 3 and Chapter 5. UT-Interaction and TV-Human dataset have been used in Chapter 4 for action detection and action prediction experiments.

Chapter 3

Fast Action Detection in Unconstrained Videos

3.1 Introduction

Human action detection is essentially one of the most important problems in the computer vision community with numerous applications in video surveillance and intelligent human computer interaction. Despite extensive studies in human action recognition and classification [3, 4, 6], detection and accurate spatio-temporal localization of human actions remains a challenging problem. Different from action classification, which only requires identifying the action type of a video clip, action detection needs to identify not only the occurrences of a specific type of actions, but also where (spatial location in each frame) and when (temporal location) it occurs in the video [9, 10, 11, 12]. One example is the detection of a person waving hands in a crowded and dynamic scene.

To robustly detect human actions, some early methods rely on the detection and tracking of human bodies. With an accurate tracking of a human body and its move-

ments, one can detect and recognize actions. These methods, however, are of limited use in practical applications, because reliable body tracking remains a challenging problem in the crowded and dynamic scenes. For example, in a supermarket with many people, it is very difficult to detect and track all the people, let alone to recognize their actions, e.g. someone raising his/her hands.

Instead of tracking human bodies, some other methods treat videos as spatio-temporal three-dimensional data, and perform action detection by a spatio-temporal template matching. Similar to the sliding window based object detection, given an action template, the re-concurrences of the query action can be found by enumerating all of the possible video sub-volumes as shown in Fig. 3.1. A detailed review has been presented in Chapter 2.3. Despite previous success of this approach, there are still two major challenges.

Figure 3.1: An illustration of sub-volume search.

First of all, for the template matching method, only a single template is usually

provided to perform action detection [26, 46]. In such a case, a single template cannot well characterize the large intra-class variations of an action and therefore is not discriminative enough for classification. Second, different from object detection, the search space in the spatio-temporal video space is extremely large. It thus greatly increases the computational cost for these template based approaches. For example, it is very time consuming to search actions from different spatial scales and different temporal durations in the video space. Although the recently proposed spatio-temporal branch-and-bound search method [9, 8] can significantly improve the search speed, it is still not fast enough to handle high-resolution videos (e.g. 320×240 and higher). Considering that the spatial localization is computationally more intensive for high resolution videos, it is important to provide efficient solutions with low computational cost for high-resolution videos. Moreover, given a video dataset containing multiple action instances, it is desirable to efficiently detect all of them in one round of search.

To address the above challenges in action detection, a random forest based template matching method is developed for action detection, as shown in Fig. 3.2. Without performing background subtraction and human body tracking, each video sequence is characterized by a set of spatio-temporal interest points (STIPs). During the training phase, a random forest is constructed to model the distribution of the STIPs from both positive and negative classes in the high-dimensional feature space. During the test phase, each individual point matches the query class through the pre-built random forest, and provides an individual voting score toward each action category. Following the mutual information maximization formulation in [9], action detection becomes finding the spatio-temporal video sub-volume which maximizes the total mutual information score.

Compared with the nearest neighbor based matching method in [9], the proposed random forest based approach enables a much more efficient interest point matching

without degrading the matching quality. Meanwhile, as both positive and negative action samples are taken into account while building the random forest, the proposed method not only handles intra-class action variations well, but also provides more discriminative matching to detect action instances. To reduce the computational overhead in searching through high resolution videos, the original spatio-temporal branch-and-bound search method in [9] is improved on two aspects: first of all, instead of performing branch-and-bound search in the original score volume, searching the down-sampled score volume for efficient action localization is proposed. The theoretical analysis shows that the error between the optimal solution of the down-sampled volume and that of the original volume can be upper bounded. Secondly, a top-K search method is proposed to enable the detection of multiple action instances simultaneously in a single round of branch-and-bound search. It provides an efficient solution for multi-class multiple instance action detection.

To evaluate the efficiency and generalization ability of the proposed method, a cross-dataset action detection test has been performed: the proposed algorithm is trained on the KTH dataset and tested on the MSR action dataset II, which contains 54 challenging video sequences from both indoor and outdoor scenes. The extensive multi-class action detection results show that, ignoring the feature extraction cost, the proposed method can search a one-hour 320×240 video sequence in less than half an hour. It can detect actions of varying spatial scales, and can well handle the intra-class action variations including performing style and speed variations, and even with partial occlusions. It can handle cluttered and dynamic backgrounds as well. The proposed Top-K volume search algorithm is general and can be utilized for any other applications of video pattern search.

Figure 3.2: The overview of the proposed random forest based action detection algorithm.

3.2 Mutual Information based Formulation

Action is represented as a collection of space-time interest points (STIPs) [1], where $d \in \mathbb{R}^N$ denotes an N -dimensional feature vector describing a STIP. The reasons to represent the videos with STIP are its superior performance over other representations and that the HoG&HoF description is suitable for the proposed random forest based framework. A comparison of different detectors and descriptors can be seen in [54]. Refer the class label set as $\mathcal{C} = \{1, 2, \dots, C\}$, where C is the number of classes.

To recognize the action category, the pointwise mutual information [50] between a

testing video clip $\mathcal{Q} = \{d_q\}$ and one action class $c \in \mathcal{C}$ is evaluated as:

$$\begin{aligned}\mathcal{H}(\mathbf{C} = c, \mathcal{Q}) &= \log \frac{P(\mathcal{Q}|\mathbf{C}=c)}{P(\mathcal{Q})} \\ &= \log \frac{\prod_{d_q \in \mathcal{Q}} P(d_q|\mathbf{C}=c)}{\prod_{d_q \in \mathcal{Q}} P(d_q)} \\ &= \sum_{d_q \in \mathcal{Q}} \log \frac{P(d_q|\mathbf{C}=c)}{P(d_q)},\end{aligned}\tag{3.1}$$

where d_q refers to the STIP point in $\mathcal{Q}$ and let us assume d_q is independent from each other. Each $s^c(d_q) = \log \frac{P(d_q|\mathbf{C}=c)}{P(d_q)}$ is the point-wise mutual information between a STIP point d_q and a specific class c .

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

$$s^c(d_q) = \mathcal{H}(\mathbf{C} = c, d_q) = \log \frac{C}{1 + \frac{P(d_q|\mathbf{C} \neq c)}{P(d_q|\mathbf{C}=c)}(C-1)},\tag{3.2}$$

where C is the number of classes. The likelihood ratio in Eq. 3.2 is computed as:

$$\frac{P(d_q|\mathbf{C} \neq c)}{P(d_q|\mathbf{C} = c)} \approx \lambda^c \exp^{-\frac{1}{\sigma^2}(\|d_q - d_{NN}^{c-}\|^2 - \|d_q - d_{NN}^{c+}\|^2)},\tag{3.3}$$

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

Despite its good performance, there are two limitations in Eq. 3.3:

1. In order to compute the likelihood ratio in Eq. 3.3, searching the nearest neighbors d_{NN}^{c+} and d_{NN}^{c-} are needed. Although locality sensitive hash (LSH) has been employed for fast nearest neighbor search, it is still time consuming for a large dataset with high-dimensional feature.
2. Only two STIPs are used to approximate the likelihood ratio in Eq. 3.3, which is usually not accurate.

To address the two problems, let us reformulate the voting score $s^c(d_q)$ in Eq. 3.2

as:

$$\begin{aligned}
 s^c(d_q) &= \mathcal{H}(\mathbf{C} = c, d_q) = \log \frac{P(d_q|\mathbf{C}=c)}{P(d_q)} \\
 &= \log \frac{P(\mathbf{C}=c, d_q)}{P(\mathbf{C}=c)P(d_q)} \\
 &= \log \frac{P(\mathbf{C}=c|d_q)}{P(\mathbf{C}=c)} \\
 &= \log P(\mathbf{C} = c|d_q) - \log P(\mathbf{C} = c).
 \end{aligned} \tag{3.4}$$

As $P(\mathbf{C} = c)$ is a constant prior, the problem boils down to calculating the posterior $P(\mathbf{C} = c|d_q)$. To enable an efficient computation, let us approximate the probability with a tree structure.

3.3 Random Forest based Voting

Random forest was first proposed to address the classification problem [29]. Later, it has been extended to handle regression problems and is used for many multimedia and computer vision applications, like [5, 30, 31, 32, 6, 41, 51]. In this chapter, random forest is employed to estimate the posterior probability $P(\mathbf{C} = c|d_q)$ in Eq. 3.4.

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

Two kinds of descriptors for STIP: HoG and HoF, are used to construct the random forest. In the following, let us first describe how to build a single decision tree, and then construct a forest by M independent trees. Assume there are N STIP points in the training set, defined as $\{(x_i, y_i), i = 1, 2, \dots, N\}$, where $x_i = (x_i^1, x_i^2)$; $x_i^1 \in R^{72}$ and $x_i^2 \in R^{90}$ refer to the HoG feature and HoF feature, respectively; $y_i \in \mathcal{C}$ is the label of the STIP (if we want to detect actions from category $\mathbf{C} = c$, let us consider STIPs with $y_i = c$ as positive examples and other STIPs as negative examples). In order to

build a tree and split the training set, a random number $\tau \in \{1, 2\}$ is first generated to indicate which kind of feature to use for splitting ($x_i^{\tau=1}$ refers to HoG feature and $x_i^{\tau=2}$ refers to HoF feature). Then two more random integer numbers e_1 and e_2 are generated, indicating the dimension indices of either HoG or HoF feature. After that, a “feature difference” can be evaluated with $D_i = x_i^\tau(e_1) - x_i^\tau(e_2), i = 1, 2, \dots, N$. For each x_i , assign it to the left child node if $x_i^\tau(e_1) - x_i^\tau(e_2) \geq \theta$ or right child node if $x_i^\tau(e_1) - x_i^\tau(e_2) < \theta$.

The threshold θ is selected to minimize the binary classification error:

$$\theta^* = \operatorname{argmin}_\theta (\min\{ \mathcal{E}(c)^L + \mathcal{E}(\bar{c})^R, \mathcal{E}(c)^R + \mathcal{E}(\bar{c})^L \}), \quad (3.5)$$

where:

$$\begin{aligned} \mathcal{E}(c)^L &= \sum_{i=1}^N I(y_i \neq c) I(x_i^\tau(e_1) - x_i^\tau(e_2) \geq \theta) \\ \mathcal{E}(c)^R &= \sum_{i=1}^N I(y_i \neq c) I(x_i^\tau(e_1) - x_i^\tau(e_2) < \theta) \\ \mathcal{E}(\bar{c})^L &= \sum_{i=1}^N I(y_i = c) I(x_i^\tau(e_1) - x_i^\tau(e_2) \geq \theta) \\ \mathcal{E}(\bar{c})^R &= \sum_{i=1}^N I(y_i = c) I(x_i^\tau(e_1) - x_i^\tau(e_2) < \theta). \end{aligned} \quad (3.6)$$

In Eq. 3.6, $I(x)$ is an indicator function: $I(x) = 1$ if $x = 1$ and 0 otherwise. And c is the action category to detect. The first two terms refer to the misclassification error of the left node and right node, respectively, when the labels of the nodes are both c . The last two terms refer to the misclassification error of the left node and right node, respectively, when the labels of the nodes are not c .

The above three parameters (τ, e_1 and e_2) can be combined as a single “hypothesis”. For instance, a hypothesis can be generated to split the dataset using the following three steps:

- Generate $\tau \in \{1, 2\}$ to indicate the feature type to use.
- Generate the dimension index e_1 and e_2 and compute the feature difference $D_i = x_i^\tau(e_1) - x_i^\tau(e_2), i = 1, 2, \dots, N$.

- Split the dataset into two sets based on a threshold on feature difference and obtain a misclassification error.

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

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

$$P(\mathbf{C} = c|d_q) \approx \frac{1}{M} \sum_{i=1}^M \frac{N_i^+}{N_i^+ + N_i^-}. \quad (3.7)$$

Then Eq. 3.4 can be replaced with:

$$\begin{aligned} S^c(d_q) &= \log P(\mathbf{C} = c|d_q) - \log P(\mathbf{C} = c) \\ &= \log \frac{1}{M} \sum_{i=1}^M \frac{N_i^+}{N_i^+ + N_i^-} - \log P(\mathbf{C} = c). \end{aligned} \quad (3.8)$$

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

The benefits of using the random forest are several-folds. First of all, each tree in the forest is independent to others when evaluating $P(\mathbf{C} = c|d_q)$ in Eq. 3.7. The average of them thus reduces the variation of the estimation. Secondly, random forest

is fast to evaluate during the test stage. The runtime cost for each STIP only depends on the depth of each tree and the number of trees. It is not affected by the number of interest points in the training data. Hence, it is much more efficient than LSH based nearest neighbor search. In the experiment section, it will show that random forest based voting approach is over 4000 times faster than the LSH based approach. Another advantage of random forest compared with LSH is that, when constructing the trees, the label information of x_i can be integrated. Thus, the trees follow the data distribution of the training data. This improves the generalization ability. Finally, the construction of random forest is flexible. Besides the label information, it is easy to integrate other types of feature descriptors and spatial information of STIPs.

In literature, the methods in [51, 6, 52] also employ tree structure for action recognition. Let us first consider the differences between [51] and this work. The feature used in [51] is densely sampled while the proposed algorithm uses the sparse STIP features. Second, in [51], it votes for the center of the action while the proposed random forest weighs each STIP point so that the non-trivial scale estimation can be partially solved with branch and bound search. Third, the votes in [51] are estimated from the frequency view so that it would generate positive votes even for the background. On the contrary, the votes employ the mutual information based measure (Eq. 3.4), which is more discriminative thanks to the introduction of negative votes. The trees in [6] are used for indexing and searching nearest neighbor while trees in [52] serves as a codebook. Since random forest is employed to weigh each STIP points, the motivations and implementations are different from [6, 52]. Besides, this work can deal with not only action classification but also action detection, while the methods in [6, 52] are only applicable to action recognition.

After obtaining the individual voting score of each STIP, the spatio-temporal location and scale of the target action will be determined by a branch-and-bound search

as described in next section.

3.4 Action Detection and Localization

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

$$V^* = \operatorname{argmax}_{V \in \mathcal{V}} f(V), \quad (3.9)$$

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

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

$$F(W) = \max_{T \subseteq \mathbb{T}} f(W \times T), \quad (3.10)$$

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

One advantage of separating the parameter space is that the worst case complexity is reduced from $O(m^2 n^2 t^2)$ to $O(m^2 n^2 t)$. The complexity is linear in t , which is usually

the largest of the three dimensions. For this reason, it is efficient in processing long videos, but when the spatial resolution of the video increases, the complexity goes up quickly. The method in [9] was tested on videos with low resolution (160×120). In this chapter, higher resolution videos (320×240 , 640×480 , or above) are processed. Videos taken under challenging lighting conditions with crowded background such as those in the publicly available MSR Action dataset II ¹, the action detection rate on 320×240 resolution videos are much better than those on 160×120 . Unfortunately, the subvolume search for 320×240 videos is much slower. For example, the method in [9] takes 20 hours to search the MSR Action dataset II which consists of 54 video sequences of 1 minute long each.

Moreover, in [9], the multi-instance detection problem was considered as a series of single subvolume search problem. They first find the optimal subvolume V_1 such that $f(V_1) = \max_V f(V)$. After that, it sets the scores of all the points in V_1 to 0, and finds the optimal subvolume V_2 , and so on. To further speed up the search process during the branch-and-bound iterations, a heuristic was used in [8]. If a candidate window $\mathbb{W}$ with a score larger than the detection threshold is found, the subsequent searches are limited to the subwindows contained in $\mathbb{W}$. It guarantees that it will find a valid detection, but the detected subvolume is not guaranteed to be optimal.

In the next two subsections, several techniques are presented to speed up the subvolume search algorithm, which allow us to perform subvolume search on 320×240 videos in an efficient way.

¹ The MSR action dataset II is available at <http://research.microsoft.com/en-us/um/people/zliu/ActionRecoRsrc/default.htm>

3.4.1 Spatial Down-Sampling

To handle high-resolution videos, a novel technique to spatially down-sample the video size by a scale factor s before the B&B search is presented. Note that the proposed algorithm extracts the local interest point and descriptor from the original video space.

Given a volume $\mathcal{V}$ with scale size $m \times n \times t$, the size of the volume after down-sampling $\mathcal{V}^s$ with scale factor s is $\frac{m}{s} \times \frac{n}{s} \times t$. Given a point $(i, j, u) \in \mathcal{V}^s$ where $i \in [0, \frac{m}{s} - 1]$, $j \in [\frac{n}{s} - 1]$, and $u \in [0, t - 1]$, $f^s(i, j, u)$ is defined as the summation of all the scores of the $s \times s$ points in $\mathcal{V}$:

$$f^s(i, j, u) = \sum_{x=0}^{s-1} \sum_{y=0}^{s-1} f(s * i + x, s * j + y, u). \quad (3.11)$$

For any sub-volume $V^s = [L, R] \times [T, B] \times [S, E] \subset \mathcal{V}^s$, where L, R, T, B, S and E are the left, right, top, bottom, start and end positions of V^s , respectively, refer $\xi(V^s)$ as the corresponding sub-volume in original video $\mathcal{V}$, i.e.,

$$\xi(V^s) = [s * L, s * (R + 1) - 1] \times [s * T, s * (B + 1) - 1] \times [S, E]. \quad (3.12)$$

As they are the same sub-volume, then:

$$f^s(V^s) = f(\xi(V^s)). \quad (3.13)$$

Let us call a subvolume $V = [X_1, X_2] \times [Y_1, Y_2] \times [T_1, T_2] \subset \mathcal{V}$ *s-aligned* sub-volume if the following conditions are satisfied: 1) both the window width $X_2 - X_1 + 1$ and height $Y_2 - Y_1 + 1$ are also multiples of s ; 2) X_1 and Y_1 are multiples of s . Eq. 3.12 provides a one-to-one mapping between the volumes in $\mathcal{V}^s$ and the s -aligned subvolumes in $\mathcal{V}$.

Instead of searching the original video space, let us search the down-sampled video space $\mathcal{V}^s$ of a much smaller size $\frac{m}{s} \times \frac{n}{s} \times t$. However, as the down-sampling process also introduces the approximation errors, it adversely affects the search results. In general,

Figure 3.3: Approximation of the spatial down-sampling. Left figure shows the score image in the original resolution and right figure shows the down-sampled score image.

for any $V^s \subset \mathcal{V}^s$, there exists a $V = \xi(V^s) \subset \mathcal{V}$. It thus shows that the maximum subvolume found in the down-sampled space is at most as good as the one found in the original space:

$$\max_{V^s \subset \mathcal{V}^s} f^s(V^s) \leq \max_{V \subset \mathcal{V}} f(V). \quad (3.14)$$

Fig. 3.3 provides a concrete example. For simplicity, in Fig. 3.3, the down-sampling factor is set as $s = 2$ and the problem in the $2D$ space (only one frame is considered) is discussed. The left figure shows the original video space and its down-sampled version is in the right figure. Each pixel is associated with a voting score. Every four small pixels in a cell from the original resolution sum up to one score in the low resolution, for example, the value in the top-left pixel from the right figure $0.5 = 0.1 + 0.3 - 0.4 + 0.5$. The orange rectangle highlights the optimal solution in the original video space, namely the bounding box of the highest total sum. After the down-sampling, the grey rectangle is the detection result in the down-sampled video. By mapping it back to the original space, an approximate solution highlighted by the red rectangle is obtained. It overlaps with the optimal solution in the original space, but the total sum of the approximated one is slightly less than optimal solution. To further quantify the approximation error, the upper bound of the error caused by the down-sampling is derived, as explained in Theorem 3.1.

Theorem 3.1. *Bound of the approximation error*

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

$$f(\tilde{V}) \geq (1 - \frac{s * h + s * w + s^2}{wh})f(V^*). \quad (3.15)$$

The proof of this theorem is in the appendix.

Let $\tilde{V}^* = \operatorname{argmax}_{V \in \mathcal{V}^s} f^s(V)$ denote the optimal subvolume in $\mathcal{V}^s$. Based on Eq. 3.15, we have:

$$f^s(\tilde{V}^*) \geq (1 - \frac{s * h + s * w + s^2}{wh})f(V^*). \quad (3.16)$$

For instance, given the spatial window $V = 320 \times 240$ and $s = 8$, then the down-sampled volume size is 40×30 . Consider an example with the optimal sub-volume window size V^* is 64×64 . So we have:

$$\frac{s * h + s * w + s^2}{wh} = \frac{8 * 64 * 2 + 64}{64^2} \approx 25\%. \quad (3.17)$$

Numerical experiments have been run to measure the relative error of the optimal solutions in the down-sampled volumes. 30 video sequences of resolution 320×240 from three action categories are used. The score defined in Eq. 3.8 can be obtained for each action category and video sequence. The scale factor s is set as 8 and spatial volume size is down-sampled to 40×30 . From the 113 actions, its corresponding down-sampled sub-volume is first obtained and then the relative error is computed. It can be found that the mean is 23% and the standard deviation is 26% and therefore the numerical experiment is consistent with the theoretical analysis.

3.4.2 Top-K Search and λ Search

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

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

λ Search

In this section, let us describe an algorithm that finds all of the subvolumes with scores larger than a user-specified threshold λ . The pseudo-code of the algorithm is shown in Algorithm 1. Similar to the notation used in [9], $\mathbb{W}$ denotes as a collection of spatial windows defined by four parameters which represent the ranges for left, right, top, and bottom positions. $\hat{F}(\mathbb{W})$ is referred to as the upper bound given any set of window $\mathbb{W}$. Denote W_{max} as the largest window among all the windows in $\mathbb{W}$. Initially, $\mathbb{W}^*$ is equal to the set of all the possible windows on the image and F^* is the corresponding

upper bound, as in Line 5 of Algorithm 1. From Lines 6-19, the results are split and stored if the top state $\mathbb{W}$ is over a threshold λ and iterate this process. From Lines 20-22, there will be a sub-volume (V^*) detected. The whole process iterates until the score for the detected sub-volume is below the threshold.

λ -search.

```

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

```

Algorithm 1: λ -search.

In terms of the worst case complexity, the number of branches of this algorithm is no larger than $O(n^2m^2)$ since the algorithm does not re-start the priority queue P . Each time it branches, the algorithm has to compute the upper bound whose complexity is $O(t)$. Therefore the worst complexity involved in branch and bound is the same as [9]: $O(n^2m^2t)$. In addition, each time when it detects a subvolume, the algorithm has to update the scores of the video volume which has complexity $O(tmn)$. Suppose

k sub-volumes are detected, the complexity for computing the scores is $O(ktmn)$. In general, the complexity for worst case is $O(n^2m^2t) + O(kmnt)$. When k is large, this is much better than $O(kn^2m^2t)$.

Top-K Search

In this section, let us describe how to modify Algorithm 1 for the case when finding the Top-K actions are interested, and assume the threshold λ unknown.

The pseudo-code of the algorithm is shown in Algorithm 2. The algorithm is similar to Algorithm 1. In Line 6, $(\{W_i^*, F_i^*\})_{i=c\dots k}$ are set for all the possible windows on the image and its upper bound score, respectively. From Lines 6-20, the results are split and stored if the top state $\mathbb{W}$ is over the Kth top score and iterate this process. From Lines 21-24, there will be a sub-volume (V^*) detected. The whole process iterates until K sub-volumes are detected. There are four major differences. First, instead of maintaining a single current best solution, it maintains k -best current solutions. Second, it replaces the criteria $\hat{F}(\mathbb{W}) > \lambda$ with $\hat{F}(\mathbb{W}) > F_k^*$ to determine whether $\mathbb{W}^1$ or $\mathbb{W}^2$ is needed to insert into the queue P . Third, it replaces the inner-loop stop criteria $\hat{F}(\mathbb{W}) \leq F^*$ with $\hat{F}(\mathbb{W}) \leq F_c^*$. Finally, the outer-loop stop criteria $\hat{F}(\mathbb{W}) \leq \lambda$ is replaced with $c > k$. In this algorithm, the number of outer loops is k . So the worst case complexity is also $O(n^2m^2t) + O(kmnt)$.

3.5 Experiments

3.5.1 Action Classification

To evaluate the proposed random forest based approach for multi-class action classification, the benchmark KTH dataset is used. The experimental setup is the same

Top-K Search.

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

Algorithm 2: Top-K Search.

as [1, 9] where clips from 16 persons are used for training and the other 9 persons are used for testing. The confusion matrix is presented in Table 3.1. The state-of-the-art results are compared in Table 3.2. With the same input features, this method performs as well as the method using support vector machine for classification [4]. Although the performance is slightly worse than the nearest neighbor based classification in [9], as will be shown later, the proposed approach is significantly faster as it avoids the nearest neighbor search.

3.5.2 Action Detection

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

For all of the experiments, $K = 3$, $\lambda = 3.0$ are fixed. Moreover, unless explicitly

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

Table 3.1: Confusion matrix for KTH action dataset. The total accuracy is 91.8%.

mentioned the score volume is down-sampled to 40×30 pixels.

Fig. 3.4 compares the precision-recall for the following methods (the original videos are of high resolution 320×240):

- (i) ASTBB (Accelerated Spatio-Temporal Branch-and-Bound search) of [8] in low resolution score volume (frame size 40×30),
- (ii) ASTBB of [8] in 320×240 videos,
- (iii) multi-round branch-and-bound search of [9] in low-resolution score volume (frame size 40×30),
- (iv) Top-K search at original size 320×240 ,
- (v) Top-K search at down-sampled score volume (size 40×30),
- (vi) λ -search at down-sampled score volume (size 40×30),
- (vii) random forest based weighting followed by Top-K search at down-sampled score volume (size 40×30).

Except for (vii), which uses the proposed random forest based voting score, the other methods employ the LSH based nearest neighbor voting score as in [9]. The parameter $A = -\log P(C = c)$ in Eq. 3.8 for method (vii) is set to 2.1, 1.7 and 0.9 for handclapping, handwaving and boxing respectively. Also, the walking actions from

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

Table 3.2: Comparison of different reported results on KTH dataset.

KTH are used as the negative dataset when constructing forests. For the purpose of generating precision-recall curves, the outer-loop stop criteria (line 25, algorithm 2) is modified to repeat until $\hat{F}(\mathbb{W}) \leq \lambda$ where λ is a small threshold. In this way, it outputs more than K subvolumes which is necessary for plotting the precision-recall curve. Some sample detection results obtained by the proposed approach (vii) are shown in Fig. 3.6. The cyan dash regions are the ground truths. The first column are sample images from training set. To demonstrate the capability of handling non-stationary actions, a walking detection result is shown at the bottom row. The detection is done by using KTH walking as the positive training data while the KTH handwaving, handclapping, and boxing are used as the negative training data.

Following the same evaluation precision-recall measure used in [9], a positive detection is considered if: $\frac{\text{Volume}(V^* \cap G)}{\text{Volume}(G)} > \frac{1}{8}$ where V^* is the detected sub-volume and G is the ground truth sub-volume. Similarly, to compute the recall, let us say a hit if: $\frac{\text{Volume}(V^* \cap G)}{\text{Volume}(V^*)} > \frac{1}{8}$.

Let us first compare the results based on LSH voting approaches. Fig. 3.4 lists the Precision-Recall curves for the three different action classes, respectively. Fig. 3.5 shows the average precision-recall curve for the three actions. A precision-recall curve is generated by adjusting the threshold to pick the sub-volumes. From the precision-recall curves, it can be seen that although the accelerated search of [8] provides excellent results in high resolution videos, its performance on down-sampled low resolution videos is poor compared with other search schemes. Moreover, all the methods applied to the high resolution videos provide similar performance. In particular, the methods of Top-K search with branch-and-bound search at down-sampled size (v) and λ -search with branch-and-bound search at down-sampled size (vi) are among the best ones. These results justify the proposed λ -search and Top-K search algorithms. Although the branch-and-bound is performed in the down-sampled size videos, it still provides good

Figure 3.4: Precision-recall curves for action detections with different methods.

performance. However, the search speed is much faster. To compare the performance of action detection between LSH and random forest, (v) and (vii) are two search schemes with the same environment but different voting approaches. Random forest (vii) is superior to LSH (v) in handwaving but poorer in boxing. Since the boxing action is highly biased in KTH dataset (much more boxing actions are performed from right to left), it reduces the discriminative ability of the trees. For LSH, however, because it searches only one nearest positive and negative in the neighborhood, the effect of such bias can almost be ignored.

Also, from Fig. 3.4, it is easy to find that the performance of handclapping action is not as good as handwaving and boxing actions. Intuitively, handclapping action is more subtle compared with handwaving and boxing actions. Besides, it is difficult to tell the difference between the local interest points from handclapping action and boxing action based on HoG and HoF description only. Also, the clutter background (human bodies with different clothes) of the handclapping actions make the problem more difficult.

Figure 3.5: Comparisons of Average Precision-Recall curves.

3.5.3 Computational Cost

The feature extraction step is performed with publicly available code in [1]. For a fair comparison, it is the same implementation which are used in [9, 14]. Therefore, the computation time for feature extraction is not considered in this chapter. Suppose all the STIP points are already extracted and stored in the memory. Then, the computational time of the proposed algorithms is dominated by two operations, computing the score for each STIP and branch-and-bound search. For the first part, LSH takes on average $18.667ms$ per STIP point while random forest only takes $0.0042ms$. To deal with a video clip with 10000 STIPs, it will take around $186.67s$ for LSH but only $42ms$ for random forest. That is, random forest based approach is 4000 times faster than LSH based approach.

Table 3.4 shows the time consumed for the search part. C++ is employed to implement the algorithms. The following algorithms are evaluated on a normal Windows-7 PC with 4G main memory:

Figure 3.6: Detection results (Random Forest+Top-K) of handclapping (1st row), handwaving (2ed row), boxing (3rd row) and walking (4th row) are listed.

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

Table 3.3: Time consumed for voting one STIP and one video sequence (for example, 10000 STIP points). Only CPU time is considered.

Method	Computational Cost (mins)
Low resolution (40×30) [8]	40
High resolution (320×240) [8]	1200
Our Down-sampled (40×30)	600
Our Down-sampled (40×30) + λ -search	80
Our Down-sampled (40×30) + Top-K	26
Our Down-sampled (80×60) + Top-K	360

Table 3.4: A comparison with the computational cost.

- (a) λ -search version of [8] in 40×30 videos,
- (b) λ -search version of [8] in 320×240 high resolution videos,
- (c) multiple rounds of B&B search of [9] in 40×30 low-resolution videos,
- (d) combined B&B search and λ -search in 40×30 videos,
- (e) combined B&B search and Top-K search in 80×60 videos,
- (f) combined B&B search and Top-K search in 40×30 videos.

According to Table 3.4, despite of good performance in low resolution videos, the algorithm [8] works much slower in the high resolution videos. Besides as shown in Fig. 3.5, when performing on the down-sampled score volumes, the heuristic method of [8] (curve (i)) is a lot worse than the other methods. This is an indication that it is not a good idea to perform heuristic search on down-sampled score volumes. In comparison, λ -search provides much better search quality. Among all the search schemes, the fastest method is the Top-K search with branch-and-bound at down-sampled score volume of 40×30 . It takes only 26 minutes to process the 54 sequences whose total length is about one hour in total.

Finally, LSH and random forest are compared in terms of total computation time in Table 3.5, including the runtime cost for computing scores and the runtime cost for

Method	Voting Time (mins)	Search Time (mins)	Total Time (mins)
LSH+B&B [8]	271	1200	1471
LSH+Top-K (our)	271	26	297
RF+Top-K (our)	0.62	26	26.62

Table 3.5: Comparison of total time cost for action detection. Only CPU time is considered.

top-K search. For the previous method [8], it takes at least 1471 minutes to search all the actions for 54 videos in MSR11. In contrast, the total computation time of the proposed algorithm is 26.62 minutes.

3.6 Concluding Remarks

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

Chapter 4

Action Detection with Limited Training Data

4.1 Introduction

Although the proposed algorithm in Chapter 3 achieved promising results for action detection when the labeled training data is sufficient, action detection with limited training data is still a big challenge. It is impossible to train the random forest based classifier in Section 3.3 without sufficient training data. However, in many surveillance environments, it is expensive and difficult to acquire the labeled training data. For instance, detecting the fighting action in the streets is useful for security but it is difficult to collect sufficient samples to train a model. In this chapter, a novel “semi-supervised” algorithm based on Hough voting will be proposed to address the problem when the training data is limited.

Generalized Hough voting has been developed to detect both objects [137, 129, 86] and activities [129]. By leveraging the ensemble of local features, where each local

feature votes individually to the hypothesis, it 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 it can provide reliable detection in cluttered and dynamic scenes.

Despite previous successes, most current Hough-voting based detection approaches require sufficient training data to enable effective local feature matching or learn the discriminative voting of local features. For example, the method in [129] requires sufficient labeled local features to train the random forest for Hough voting. When few training examples are provided, the performance of such a method is likely to suffer due to the unreliably learned voting scores.

One key challenge of Hough voting based method is to obtain effective local voting scores for individual features, such that the summation of these scores can bring robust detection. However, in case when the training data is limited, such as searching for similar actions/activities with few examples, such situations pose great challenge to the existing Hough voting based approaches. To make local voting robust with limited training data, propagative Hough voting is proposed which can leverage the distribution of local features in the feature space for better local feature voting. To enable efficient and accurate local feature voting, random projection trees (RPT) [114] are employed to index the local features. Given a local feature, its voting score will be determined via its matching across the multiple RPTs. Compared with using a predefined distance metric to find the best match, RPT explores the data distribution of local features thus bringing the benefit of adaptive matching. Moreover, RPT enables fast matching of local features.

Fig. 4.1 briefly illustrates how the proposed method works. The left figure illustrates the proposed model on a training video which utilizes the spatio-temporal configuration of local interest points. In the right figure, three local features (STIPs) from the test

Figure 4.1: An illustration of Propagative Hough Voting.

videos are illustrated with blue triangles. In the middle figure, the three yellow dots indicate the matched STIPs from the training data. For each STIP (yellow dot) in the training video, the spatial-temporal configuration information will be propagated to the matched STIPs (blue triangle) for Hough voting in the test video. By accumulating the votes from all the matched 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 be built on either labeled data or unlabeled data. For each local patch (or feature) from the training example, it searches for the best matches through random projection trees. Once the matches in the test video are found, the label and spatial-temporal configuration information are propagated from the training data to the test data. The accumulated score obtained from Hough voting formulation can be used for recognition and detection.

The major advantages of the proposed propagative Hough voting are summarized as follows:

- By leveraging video data through random projection trees, the proposed algorithm brings robust local interest point matching and thus improves activity detection and recognition performance, especially in the case when the training

data is limited.

- Following the recognition by detection paradigm, the proposed algorithm can be easily extended to address activity recognition as well as activity prediction since the model is additive to the interest points in the test video.
- Compared with existing Hough voting based human activity recognition algorithms, e.g., [129], the proposed method does not rely on human detection and tracking for human localization. A spatial-temporal scale determination algorithm is proposed to refine the localization of activities.

Two benchmarked datasets UT-Interaction [132] and TV Human Interaction [122] are used to evaluate the proposed algorithm. For a fair comparison with existing methods, propagative Hough voting is tested for activity search, activity recognition, and activity prediction as separate tasks. The superior performances over the state-of-the-arts validate that the proposed method can well handle all the applications.

4.2 Background on Hough Voting

Hough transform is first introduced to image processing and computer vision community for line detection [74] in 1981. Later it is extended to address object detection in [137] and superior performance has been achieved. In Section 5.4.2, Hough voting will be used to refine the spatial localization of human action. Compared with traditional sliding-window based approaches [16, 26], Hough voting has computational advantages since it does not need to enumerate all the possible windows in an image (or video) as in [16, 26]. Although speed-up techniques like Branch-and-Bound [9, 20] are employed for sliding-window approaches, these extensions usually ignore the spatio (or spatio-temporal) configuration of local feature points. On the other hand,

Hough voting makes a good trade-off between the computational cost and performance by modeling local feature points with an implicit shape model [137]. In [86], Hough voting is further improved by discriminative voting weights which is learnt by a max-margin optimization. In [143], latent variables are integrated with Hough voting to enforce the consistency among the votes. In [129], random forest is used for Hough voting with efficient computational speed and promising accuracy for both object detection in images and action detection in videos. The proposed work in this chapter is similar to [129] but there are three major differences. First, the trees in this chapter can explore the test data distribution which allows propagative Hough voting to well address the limited training data problem. Second, the trees in [129] are supervisedly constructed for the classification purpose while the trees in this chapter are trying to leverage the underlying data distribution in an unsupervised way. Third, propagative Hough voting does not rely on human detection and tracking, which is employed in [129] for localizing the human actor. In this algorithm, both spatial and temporal centers of the activity can be determined by Hough voting and the scale can be further refined with back-projection. With an iterative scale refinement procedure, the proposed method can also handle small scale variations.

4.3 Propagative Hough Voting

4.3.1 Problem Formulation

To characterize videos, similar as in Chapter 3, spatial-temporal interest points (STIP) [1] are first extracted from each video. Each STIP is described with Histogram of Gradient and Histogram of Optical Flow. In total, the feature dimension is 162. Each training video is represented with the spatio-temporal configuration of

STIPs as shown in Fig. 4.1. Although only sparse STIP feature is applied in the experiments, the proposed method is also applicable to dense local features. Denote $\mathcal{R} : \{d_r = [\mathbf{f}_r, l_r]; r = 1, 2, \dots, N_{\mathcal{R}}\}$ as the training data, where $\mathbf{f}_r$ and l_r is the descriptor and 3D location of interest point d_r , respectively. $N_{\mathcal{R}}$ is the total number of interest points.

Suppose a set of test videos (e.g., database videos in activity search) are provided, denoted by $\mathcal{S} = \{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_{N_{\mathcal{S}}}\}$, the goal is to recognize, locate or predict the specific activity in $\mathcal{S}$ given the training video set $\mathcal{R}$ (e.g., query video in activity search), where $\mathcal{R}$ can be one or multiple training examples. The goal is to find a video sub-volume, V^* , by matching the following objective function:

$$V^* = \arg \max_V S(V(\mathbf{x}, t, \rho), \mathcal{R}), \quad (4.1)$$

where $V(\mathbf{x}, t, \rho)$ refers to the sub-volume in $\mathcal{S}$ with temporal center t and spatial center $\mathbf{x}$; ρ refers to the scale size and duration; $S(\cdot, \cdot)$ is the similarity measure between a video sub-volume V and the whole training data $\mathcal{R}$. In total, there are 6 parameters (center position x , y , t , and width, height, duration) to locate the optimal sub-volume V^* .

To measure the similarity between $V(\mathbf{x}, t, \rho)$ and the training data $\mathcal{R}$, implicit spatial-temporal shape model (i.e., utilizing the spatio-temporal configuration information as Fig. 4.1), an extension of 2D implicit shape model in [137], is proposed to address the matching problem.

Define $S(V(\mathbf{x}, t, \rho), \mathcal{R})$ in Eq. 4.1 as follow:

$$\begin{aligned} S(V(\mathbf{x}, t, \rho), \mathcal{R}) &= \sum_{d_r \in \mathcal{R}} p([\mathbf{x}, t, \rho], d_r) \\ &= \sum_{d_r \in \mathcal{R}} p([\mathbf{x}, t, \rho] | d_r) p(d_r), \end{aligned} \quad (4.2)$$

where $d_r = [\mathbf{f}_r, l_r]$. $\mathbf{f}_r$ denotes the feature description and l_r denotes 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 , let us skip $p(d_r)$ and focus on the local feature voting $p([\mathbf{x}, t, \rho]|d_r)$:

$$\begin{aligned} p([\mathbf{x}, t, \rho]|d_r) &= \sum_{d_s \in V} p([\mathbf{x}, t, \rho], d_s|d_r) \\ &= \sum_{d_s \in V} p([\mathbf{x}, t, \rho]|d_s, d_r)p(d_s|d_r) \\ &= \sum_{d_s \in V} p([\mathbf{x}, t, \rho]|l_s, l_r)p(\mathbf{f}_s|\mathbf{f}_r), \end{aligned} \quad (4.3)$$

where d_s denotes the interest points from test sub-volume V , $\mathbf{f}_s$ denotes the feature description and l_s denotes the location of the STIP point d_s in V .

In Eq. 4.3, $p(\mathbf{f}_s|\mathbf{f}_r)$ determines the voting weight which relies on the similarity between $\mathbf{f}_s$ and $\mathbf{f}_r$. How to compute $p(\mathbf{f}_s|\mathbf{f}_r)$ will be discussed in Section 4.3.2. $p([\mathbf{x}, t, \rho]|l_s, l_r)$ determines the voting position. Suppose $d_r = [\mathbf{f}_r, l_r] \in \mathcal{R}$ matches $d_s = [\mathbf{f}_s, l_s] \in V$, spatial-temporal information from the training data is propagated to the test data with voting position $l_v = [\mathbf{x}_v, t_v]$:

$$\begin{aligned} \mathbf{x}_v &= \mathbf{x}_s - \eta_{\mathbf{x}}(\mathbf{x}_r - c_r^{\mathbf{x}}) \\ t_v &= t_s - \eta_t(t_r - c_r^t), \end{aligned} \quad (4.4)$$

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 and duration (the scale size of the test video, i.e., ρ , over the scale size of the matched training video).

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

$$p([\mathbf{x}, t, \rho]|l_s, l_r) = \frac{1}{Z} e^{-\frac{\|[\mathbf{x}_v - \mathbf{x}, t_v - t]\|^2}{\sigma^2}}, \quad (4.5)$$

where Z is a normalization constant and σ^2 is a bandwidth parameter.

4.3.2 Interest Point Matching

The matching of local features $p(\mathbf{f}_s|\mathbf{f}_r)$ plays an essential role in Hough voting. According to Eq. 4.3, as each $d_r \in \mathcal{R}$ will be matched against all $d_s \in V$, an efficient and accurate matching is critical. Random projection trees (RPT) [114], which are constructed in an unsupervised way, are employed to model the underlying low-dimension feature distribution, as the light magenta regions shown in Fig. 4.1. Random indexing trees in Chapter 5 can be utilized as well. Different from random indexing trees, random projection trees have good theoretical analysis on leveraging the underlying data distribution as claimed in [114]. That is, compared with pre-defined distance metric such as Euclidean distance, RPT can provide a more accurate evaluation of $p(\mathbf{f}_s|\mathbf{f}_r)$ with the help of underlying data distribution. However, random indexing tree is significantly faster than RPT when performing local interest point matching due to the simple implementation structure. The details of random indexing tree are given in Section 5.3.

Actually, RPT has three unique benefits compared with other data structures, e.g., [138]. First of all, as proven in [114], 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 validated by the experimental results in Section 4.5. Second, similar to BoW model, the feature space is quantized by tree structures. Rather than enumerating all the possible interest point matches, it can be efficient to 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, more accurate estimation can be made by increasing the number of trees. In the Appendix, the theorem will be proved that random projection tree based Hough voting generates optimal solution when the number of trees approaches infinity. In the following section, let us talk about

how to implement the random projection trees.

Random Projection Trees

Assume there are a set of STIPs, denoted by $\mathcal{D} = \{d_i; i = 1, 2, \dots, N_{\mathcal{D}}\}$, where $d_i = [\mathbf{f}_i, l_i]$ as defined in Section 4.3.1 and $N_{\mathcal{D}}$ is the total number of interest points. Following the standard implementation of STIP, the feature dimension is $n = 162$ (the feature dimension of HoG is 72 and the feature dimension of HoF is 90).

```

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( $x$ ) :=  $x \cdot v \leq (\text{median}(\{z \cdot v; z \in \mathcal{D}\}) + \gamma)$ 
10:  LTree  $\leftarrow$  BuildTree( $\{x \in \mathcal{D}; \text{Rule}(x) = \text{true}\}$ , depth+1)
11:  RTree  $\leftarrow$  BuildTree( $\{x \in \mathcal{D}; \text{Rule}(x) = \text{false}\}$ , depth+1)
12: end if

```

Algorithm 3: Trees = ConstructRPT($\mathcal{D}$)

Random projection trees [114] are implemented as shown in Algorithm 3. There are two parameters related to the construction of trees. N_T is the number of trees and δ_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. 4.3 will be computed as:

$$p(\mathbf{f}_s|\mathbf{f}_r) = \frac{1}{N_T} \sum_{i=1}^{N_T} I_i(\mathbf{f}_s, \mathbf{f}_r), \quad (4.6)$$

where N_T refers to the number of trees and

$$I_i(\mathbf{f}_s, \mathbf{f}_r) = \begin{cases} 1, & \mathbf{f}_s, \mathbf{f}_r \text{ in the same leaf of tree } T_i \\ 0, & \text{otherwise} \end{cases} \quad (4.7)$$

Thus, Eq. 4.2 becomes

$$\begin{aligned} S(V(\mathbf{x}, t, \rho), \mathcal{R}) & \\ & \propto \sum_{d_r \in \mathcal{R}} \sum_{i=1}^{N_T} \sum_{d_s \in V} I_i(\mathbf{f}_s, \mathbf{f}_r) p([\mathbf{x}, t, \rho] | l_s, l_r) \\ & \propto \sum_{d_r \in \mathcal{R}} \sum_{i=1}^{N_T} \sum_{d_s \in V \ \&\& \ I_i(\mathbf{f}_s, \mathbf{f}_r)=1} p([\mathbf{x}, t, \rho] | l_s, l_r), \end{aligned} \quad (4.8)$$

where $d_s \in V \ \&\& \ I_i(\mathbf{f}_s, \mathbf{f}_r) = 1$ refers to the interest points from sub-volume V which fall in the same leaf as d_r in the i th tree. Based on Eq. 4.5, the voting score can be computed as:

$$\begin{aligned} S(V(\mathbf{x}, t, \rho), \mathcal{R}) & \\ & \propto \sum_{d_r \in \mathcal{R}} \sum_{i=1}^{N_T} \sum_{d_s \in V \ \&\& \ I_i(\mathbf{f}_s, \mathbf{f}_r)=1} e^{-\frac{\|[\mathbf{x}_v - \mathbf{x}, t_v - t]\|^2}{\sigma^2}}. \end{aligned} \quad (4.9)$$

Theoretical Justification

The matching quality of Eq. 4.6 depends on the number of trees N_T . To justify the correctness of using random projection trees for interest point matching, it can be shown that when the number of trees is sufficient, the proposed Hough voting algorithm can obtain the optimal detection results. For simplicity, assume the hypothesis space is of size $W \times H \times T$, with W, H, T denoting as the width, height and duration of the test data, respectively. Each element refers to a possible center position for one activity and the scale ρ is fixed. 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, one position for each match

rather than a smoothed region in Eq. 4.5 is voted. That is,

$$p(l^*|l_s, l_r) = \begin{cases} 1, & l^* = l_v \\ 0, & \text{otherwise} \end{cases} \quad (4.10)$$

A random variable $\mathbf{x}^{(i)}$ is introduced with Bernoulli distribution to indicate whether there is a vote for the position l^* or not in the i th match. Denote q as the match accuracy and therefore $p(\mathbf{x}^{(i)} = 1) = q$. Let us introduce another random variable with Bernoulli distribution $\mathbf{y}^{(i)}$ to indicate whether there is a vote for the background position l_j (where $l_j \neq l^*$) or not in the i th match. Suppose each background position has an equal probability to be voted, then $p(\mathbf{y}^{(i)} = 1) = \frac{1-q}{N_H}$. The following theorem is proved in the appendix.

Theorem 4.1. Asymptotic property of propagative Hough voting: *When the number of trees $N_T \rightarrow \infty$, we have $S(V(l^*), \mathcal{R}) > S(V(l_j), \mathcal{R})$ with probability $1 - \Phi\left(\frac{-(q - \frac{1-q}{N_H})\sqrt{N_M}}{\sigma_{xy}}\right)$. 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 $N_T \rightarrow \infty$.*

In Theorem 4.1, $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{x^2}{2}} dx$ and σ_{xy} refers to the variance. N_M refers to the number of matches according to Eq. 4.8: $N_M = N_T \times N_{\mathcal{R}} \times N_L$ if RPT is built on the test data, and $N_M = N_T \times N_V \times N_L$ if RPT is built on the training data. $N_{\mathcal{R}}$ and N_V denote as the number of interest points in training set $\mathcal{R}$ and test sub-volume V , respectively. 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 δ_d denotes the tree depth and $N_{\mathcal{D}}$ the size of the data for building RPT. Based on the empirical simulation experiments, q is much larger than $\frac{1}{N_H+1}$. Thus the asymptotic property is true.

4.3.3 Scale Determination

In previous section, determining the activity center given the fixed scale with a Hough voting formulation has been discussed. But in some cases when the test videos are not segmented, the scale of the action needs to be determined as well. In this section, let us discuss an iterative scale refinement approach which can well address the scale issue when the test videos are not segmented.

The initial scale information ρ is set to the average scale of the training videos. Based on the Hough voting step discussed in Section 4.3.1, the rough position of the activity center can be obtained. Then back-projection, which has been used in [135, 137] for 2D object segmentation or localization, is used to determine the scale parameters.

After the Hough voting step, a back-projection score for each testing interest point d_s from the test video based on Eq. 4.2 can be computed as:

$$\begin{aligned} s_{d_s} &= \sum_{d_r \in \mathcal{R}} p(l^* | l_s, l_r) p(\mathbf{f}_s | \mathbf{f}_r) \\ &= \frac{1}{Z} \sum_{d_r \in \mathcal{R}} e^{-\frac{\|l^* - l_s\|^2}{\sigma^2}} p(\mathbf{f}_s | \mathbf{f}_r), \end{aligned} \quad (4.11)$$

where l^* is the activity center computed from last round; Z and σ^2 are, respectively, normalization constant and kernel bandwidth, which are the same as in Eq. 4.5. $p(\mathbf{f}_s | \mathbf{f}_r)$ is computed by Eq. 4.6. 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, the original sub-volume is first enlarged in both spatial and temporal domains by 10%. Denote $V_{W \times H \times T}^{l^*}$ as the extended volume, meaning a volume centered at l^* with width W , height H and duration T . The optimal sub-volume $V_{w^* \times h^* \times t^*}^{l^*}$ is searched to maximize the following function:

$$\max_{w^*, h^*, t^*} \sum_{d_s \in V_{w^* \times h^* \times t^*}^{l^*}} s_{d_s} + \tau w^* h^* t^*, \quad (4.12)$$

where τ is a small negative value to constrain the size of the volume.

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, they are considered as the background. This motivates us to use the method in Eq. 4.12 to locate the optimal sub-volume $V_{w^* \times h^* \times t^*}^{l^*}$.

Once the scale information of the sub-volume is obtained, ρ in Eq. 4.1 is replaced with $[w^*, h^*, t^*]$ computed from Eq. 4.12 and a new round of Hough voting is started. The process iterates until convergence or reaching a pre-defined iteration number.

4.4 Applications

4.4.1 Human Activity Detection

Following Eq. 4.1, a sub-volume V^* should be found in the database $\mathcal{S}$ which maximizes the similarity between V^* and training video set $\mathcal{R}$ (in human action search, there could be only one training video as the query). To estimate ρ in activity localization, the iterative refinement method as discussed in Section 4.3.3 is utilized, which iteratively applies the Hough voting and scale refinement. The reason to use the iterative algorithm is that there are 6 parameters to search for, which cannot be well handled in traditional Hough voting [137], especially when there is not sufficient amount of training data. There are two steps for the iterative refinement: 1) fix the scale, search for the activity center with Eq. 4.1; 2) fix the activity center, and determine the scale ρ based on back-projection in Section 4.3.3. The initial scale is set as the query video size. The two steps are iterated until convergence.

4.4.2 Human Activity Recognition

Similar to human action detection, the formulation in Eq. 4.1 can be easily extended for human activity recognition and prediction. Different from activity detection, the test videos for human activity recognition and prediction have already been segmented. Thus, the scales are directly from the test videos (width, height and duration) instead of computing the scale with Section 4.3.3.

The goal of human activity recognition is to determine the activity category of a segmented video clip, referred to as V , among a set of classes $\{1, \dots, K\}$. Based on Eq. 4.1, the problem is to find the category $k \in \{1, 2, \dots, K\}$ which maximizes the following function:

$$\max_k S(V(\mathbf{x}, t, \rho), \mathcal{R}_k), \quad (4.13)$$

where $\mathcal{R}_k$ refers to the training data from the k th category; $\mathbf{x}, t$ refers to the spatial-temporal center of the test video V ; ρ refers to the size of the test video. The similarity function $S(V(\mathbf{x}, t, \rho), \mathcal{R}_k)$ can be computed from Eq. 4.2. To compute Eq. 4.2, all the interest points from the training videos $\mathcal{R}$ need to be enumerated. According to Eq. 4.7, only a small number of training interest points which have matches with testing interest points will be used for voting. A lot of computational cost can be saved by only considering these training interest points which have matches ($I_i(\mathbf{f}_s, \mathbf{f}_r) = 1$ in Eq. 4.7) with testing interest points.

4.4.3 Human Activity Prediction

Activity prediction [109][108] is to predict the activity category given incomplete observations V^δ . Similar to the human activity recognition in Section 4.4.2, the problem can be formulated to find the category $k \in \{1, 2, \dots, K\}$ which maximizes the following

4.5. Experiments

Figure 4.2: Activity detection results on UT-Interaction dataset.

function:

$$\max_k S(V^\delta(\mathbf{x}, t, \rho), \mathcal{R}_k), \quad (4.14)$$

where V^δ refers to the percentage ($\delta \in [0, 100\%]$) of video $\mathcal{V}$ that has been observed.

According to Eq. 4.3, it can be seen that the formulation is additive based on the observed videos (interest points in that part of test videos). Thus, the prediction problem can be solved with the similar way as the human activity recognition. Generally, with more video contents observed, better performance can be achieved.

4.5 Experiments

UT-Interaction [132] and TV Human Interaction dataset [122] are used to evaluate the proposed algorithm. The two datasets are used to test the performance of activity

detection, activity recognition and activity prediction, respectively.

4.5.1 Experiments on Human Activity Detection

Activity Detection with Localization on UT-Interaction dataset

The activity detection algorithm is evaluated on the continuous UT-Interaction dataset. To evaluate the action detection algorithm when the training data is limited, just a few or even a single training sample which indicate what kind of activity the user wants to find are available. Following the requirement of such an application scenario, the proposed algorithm is tested with only one query sample randomly chosen from the segmented videos. But if more training samples are available to the proposed algorithm, the performance will be further boosted. With the help of the iterative activity detection algorithm, all similar activities can be efficiently located in a large un-segmented (continuous) video set. To compute the precision and recall, 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. 4.2 shows the results of different algorithms. As previous works of activity detection do not provide precision-recall curves on this dataset, only the following algorithms are compared: Branch&Bound [9, 20] (magenta curve) and nearest neighbors+Hough voting without scale determination (green curve). The same code provided by Chapter 5 is used to run the results. There are two categories of 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, it can be seen 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

Figure 4.3: Activity detection results on the UT-Interaction Dataset. For each category, the first image is from the query video and the following three images are sample detection results. The red bounding boxes enclose the detected sub-volumes.

changes, only the results after one round of iteration are presented. The performance does not improve significantly when the number of iterations is further increased.

Fig. 4.3 provides sample results of the proposed activity detection 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. 4.3. Video results can be found in the website.¹

4.5.2 Experiments on Human Activity Recognition

For the human activity recognition experiments, the proposed algorithm is able to handle standard activity recognition setting, i.e., the training data is sufficient. Besides, experiments when the training data is limited are also tested where RPT is built on the test data without labels. Superior performance on both UT-Interaction dataset and TV Human Interaction show that the algorithm outperforms the state-of-arts on both the cases when the training data is sufficient and insufficient.

Activity recognition on sufficient training data

The first experiment with sufficient training data employs the setting in the activity recognition contest [130]. It is a 10-fold leave-one-out cross validation. Table 4.1 and Table 4.2 list the published results on two different sets of videos. Since enough training data is provided, unsupervised random projection trees are built from the training data. The experimental results show that the proposed algorithm outperforms the state-of-the-art methods on the classification problem when the amount of training data is sufficient. Specifically, compared with [129], even though the random trees are not supervisedly constructed as [129], the proposed algorithm has better performance due

¹<https://sites.google.com/site/skicyyu/eccv2012activityrecognition>.

to the benefits from exploring the underlying data distribution.

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
[79] + kNN	0.56	0.85	0.33	0.93	0.39	0.72	0.63
[79] + Bayes	0.49	0.86	0.72	0.96	0.44	0.53	0.667
[79] + SVM	0.8	0.9	0.9	1	0.7	0.8	0.85
[129]	0.7	1	1	1	0.7	0.9	0.88
[109] BoW	-	-	-	-	-	-	0.85
[17] Phrases	0.8	0.8	1	0.9	0.9	0.9	0.88
Our proposed	1	1	1	1	0.6	1	0.933

Table 4.1: Comparison of classification results on UT-Interaction **Set 1** with leave-one-out cross validation setting.

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
[79] + kNN	0.65	0.75	0.57	0.9	0.58	0.25	0.617
[79] + Bayes	0.26	0.68	0.72	0.94	0.28	0.33	0.535
[79] + SVM	0.8	0.8	0.6	0.9	0.7	0.4	0.7
[129]	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

Table 4.2: Comparison of classification results on UT-Interaction **Set 2** with leave-one-out cross validation setting.

According to Table 4.1 and Table 4.2, results from cuboid features [79] are better than those from STIP features [1]. Although STIP features are used, better results than the state-of-the-art techniques that use cuboid features can be achieved.

Since UT-Interaction is recorded in controlled environments, the TV Human Dataset [122] is used to show that the proposed algorithm is also capable of handling activity detection and 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*. The standard setting in TV-Human Interaction dataset [122] is followed: training with 25 videos for each activity and testing on the remaining videos. In addition to [122], 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 only the video data is utilized.

Following the evaluation method in [122], the proposed algorithm is also evaluated based on average precision. Table 4.3 compares the results with those reported in [122]. “+Neg” means adding 100 negative videos that do not contain the target activities into the test dataset. The precision-recall curves from the proposed algorithm are shown in Fig. 4.4.

	100 Videos	100 Videos + 100 Neg
[122]	0.3933	0.3276
Our algorithm	0.6616	0.5595

Table 4.3: Comparison of activity classification on TV Human Dataset based on average precision.

Activity recognition with insufficient training data

Method	[132]	[134]	NN + HV	RPT + HV
Accuracy	0.708	0.789	0.75	0.854

Table 4.4: Comparison of recognition results on UT-Interaction (20% training).

Two different experimental settings on UT-interaction dataset are used to validate the performance of the proposed algorithm when the training data is insufficient. The first setting is 20% data for training and the other 80% for testing. This evaluation

Figure 4.4: PR curves for activity recognition on TV Human Dataset (25 videos for each activity used for training).

method has been used in [132, 134]. Since the training data is not sufficient, random projection trees are built from the test data. The results are listed in Table 4.4. “NN + HV” refers to the method that nearest neighbor search is used to replace RPT for feature points matching. It shows that the proposed algorithm has significant performance advantages compared with the state-of-the-arts.

In Fig. 4.5, the asymptotic property of the trees is evaluated. All the parameters are fixed except the number of trees. As shown in Fig. 4.5, the performance of the proposed algorithm gradually increases (with a little variation) along with the number of trees. Finally, the proposed algorithm converges when the number of trees arrives 120. This experiment empirically validates the asymptotic property in Theorem 4.1.

The second setting is training on only one video clip for each activity type and testing on the other video clips as first used in [101]. To compare with [101], the same experiments is performed with just a single video clip as the training data for each

Figure 4.5: Asymptotic experiment on UT-Interaction (20% training).

activity type. An average accuracy of 73% is obtained which is significantly better than the average accuracy of 65% as reported in [101].

4.5.3 Experiments on Human Activity Prediction

The same setting as [109] (Leave-one sequence-out cross validation) on UT-Interaction dataset [130] is used to evaluate the proposed algorithm on activity prediction. Fig. 4.6 shows the results compared with the other algorithms on set 1 and set 2, respectively. The following algorithms are compared: *dynamic BoW*[109], *integral BoW*[109], *SVM based on Cuboid features*, *Bayesian classifiers with Gaussian models*, *BP-SVM: constructing a set of SVMs for each observation level*, *Voting: a basic voting-based approach that casts a probabilistic vote for each cuboid feature*. The details of these algorithms can be referred to [109].

According to Fig. 4.6, on average 20% performance gain is obtained over the state-of-the-art techniques. Remarkably, with only 60% observations, the proposed algorithm achieves over 80% accuracy on both set 1 and set 2. This demonstrates that the

Figure 4.6: Human activity prediction on GT-Interaction dataset. (Set 1: Left; Set 2: Right)

proposed algorithm is well suited for activity prediction with incomplete observations. Although authors in [85] report better results with less than half observation on action prediction experiments, the computational cost is its major issue since it is K times (K is the number of category) slower than the proposed algorithm. Also, the memory cost is K times larger than the proposed method since it needs to supervisedly construct K sets of trees (one set for each category). More importantly, the method in [85] cannot incorporate unlabeled data in their framework, which makes it infeasible when the training set is limited.

4.5.4 Computational Complexity

Here only the online computational cost is discussed 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 4.3.2, and W', H', T' are the width, height and duration of the test 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 4.3.2 and $T \ll T'$. It takes approximately 10 seconds to perform the activity recognition for each 4-second long test video and 15 seconds for activity detection on a 1 min test 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 with Windows 7 operating system.

4.6 Concluding Remarks

Different from the algorithm in Chapter 3 which is targeted to handle the action detection with sufficient training data, this chapter presented a “semi-supervised” model called, propagative Hough voting, for the action detection when the training data is limited. By performing feature voting with random projection trees, the proposed algorithm can leverage both the labeled and un-labeled video data. Moreover, since propagative Hough voting formulation is additive to the testing interest points, it can be employed to address the challenging human activity prediction problem. Last but not least, with a scale refinement step, the proposed algorithm can effectively locate the actions with a spatio-temporal subvolume. Experiments on benchmarked datasets further validate the performance of the proposed algorithm.

Chapter 5

Interactive Action

Detection/Search using

Randomized Visual Vocabularies

5.1 Introduction

Action search can be considered as one extreme case of action detection where only one query sample is provided to search the similar actions. Different from the action detection problem which have been well addressed in Chapter 3 and Chapter 4, the problem of action search has three unique challenges.

First of all, for action search, usually only a single query example is provided similar to content based image retrieval. In such a case, the amount of training data is extremely limited and only available at the time of query, whereas in action classification [3, 18] and detection [14, 19], both positive and negative training examples can be leveraged. Therefore it is much more difficult to identify and locate a specific

Figure 5.1: Overview of the proposed algorithm.

action example in videos. Furthermore, possible action variations such as scale and view point changes, style changes and partial occlusions only worsen the problem, let alone cluttered and dynamic backgrounds.

Secondly, a video search engine must have a fast response time because otherwise the user experience would suffer. Unlike video event recognition [37], where the goal is to rank pre-segmented video shots, action search is much more difficult as it not only needs to recognize the target action, but also locate it accurately in spatio-temporal video space, i.e., identify the spatio-temporal location of the action in the video. The accurate localization is of great challenge especially for the videos of dynamic or crowded scenes. Different from the image search, a video, e.g., a Hollywood movie or a surveillance video, usually contains millions of frames. It is worthless to return a whole video without any temporal localization. Especially for the surveillance cases, a

lot of manual work is required to determine when and where an invasion or abnormal happens. Furthermore, indexing and mapping the database is also a time-consuming process. In general, for a dataset consisting of tens of hours of videos, such an action search process is expected to be finished in only a few seconds.

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

An unsupervised action search engine will be built in this chapter to address the above three challenges. An overview of the proposed system is depicted in Fig. 5.1. Each video is characterized by a collection of interest points, which will be labeled according to the exemplar action in the query phase. The spatio-temporal video sub-volumes are cropped out as detections, if most of the interest points inside them match well with the query action.

There are three major contributions of this chapter. First of all, the concept of randomized visual vocabularies is introduced to handle the large in-class variances problem. The idea under the multiple copies of the visual vocabularies is that one vocabulary is only one view of the data and is not sufficient to retrieve all the similar actions under different views. With the help of multiple views and representations from different copies of visual vocabularies, the large in-class variances problem could be largely relieved. To implement the randomized visual vocabularies, random indexing trees are proposed with efficient speed and superior performance. Different from random forest in Chapter 3, random indexing trees are constructed in an unsupervised way for local interest point indexing. Compared with random projection tree in Chapter 4, the proposed random indexing tree is significantly faster for local interest pointing

indexing and matching. Second, the proposed coarse-to-fine subvolume search strategy significantly improves the efficiency of the state-of-the-art action detection methods, with comparable search performance. With a single desktop computer, the proposed method can search 5-hour long video within only 37.6 seconds, which significantly speed up the Top-K branch-and-bound search in Chapter 3 and propagative Hough voting in Chapter 4. Finally, the proposed method can be easily extended to support interactive search by incrementally adding user labeled actions to the query set. Experiments on cross-dataset search validate the effectiveness and efficiency of the proposed method.

5.2 Randomized Visual Vocabularies for Video Representation

Following the similar video representation in Chapter 3 and Chapter 4, an action is represented by a set of spatial-temporal interest points (STIP) [1], denoted as $V = \{d_i \in \mathbb{R}^n\}$. Each STIP point d is described by two kinds of features: HoG (Histogram of Gradient) and HoF (Histogram of Flow) and the feature dimension is 162 ($n = 162$). For action retrieval, a database with N video clips is provided (each video clip is denoted as $\mathcal{V}_i$), $\mathcal{D} = \{\mathcal{V}_1 \cup \mathcal{V}_2 \cup \dots \cup \mathcal{V}_N\}$. These video clips may contain various types of actions such as handwaving, boxing, and walking and last for several hours long.

In order to fast search human action in the large database, indexing becomes one of the most crucial parts. Traditionally, Bag-Of-Words models [4] with hierarchical K-means is widely used for interest points indexing. However, there is usually only one copy of the vocabulary. One vocabulary can only represent one view of the data. There may be two problems in that case. One is that the matched points are not reliable since the two points may only match in this view but not in other views. Usually,

multiple matches in different views can increase the reliability of the precision. The other problem is that it may miss a lot of similar samples. Finally, it would lead to low recall of the system. Even though it works fine for most of the classification jobs, it is difficult to handle the large in-class variances problems in the action search without sufficient training data as in action classification. Thus, multiple copies of vocabularies, which can well handle the above two problems, are introduced. To make each vocabulary independent, randomness should be injected to each vocabulary.

To implement the vocabularies, one solution is locality sensitive hashing. With multiple copies of hashing functions, it is easy to determine the matches. But according to the experimental results (Table. 5.2), searching the approximated nearest neighbors with LSH is a extreme time-consuming step. In order to reduce the computation cost, tree structure is employed. Although there have been a lot of works on the tree structures for computer vision applications, little work has been done for efficient index with tree structures. KD-tree allows exact NN search but it is inefficiency in high dimension cases and only slightly better than the linear search at the worst cases. Hierarchical K-means usually leads to un-balance trees and training is a time consuming process. Tree balance is defined on the root node of a tree. If the depth of left tree branch is of the similar size as the depth of right tree branch, then we call this tree balanced. Otherwise, the tree is considered as unbalanced. To overcome the above problems, the random indexing trees are proposed, which can explorer the data distribution in the high dimension cases and index the database in an efficient and effective way. As shown in Fig. 5.8, the proposed random indexing trees even slightly outperform the LSH based vocabularies.

Assume there are N_D STIP points in the dataset, denoted as $\{x_i = (x_i^1, x_i^2), i = 1, 2, \dots, N_D\}$; $x_i^1 \in \mathbb{R}^{72}$ and $x_i^2 \in \mathbb{R}^{90}$ are the HoG feature and HoF feature, respectively. In order to build a tree and split the dataset, a random number $\tau \in \{1, 2\}$ is

first generated to indicate which kind of feature to use for splitting ($x_i^{\tau=1}$ refers to HoG feature and $x_i^{\tau=2}$ means HoF feature.) Then two more random numbers e_1 and e_2 will be generated which are the dimension indices of the feature descriptor (either HoG feature or HoF feature depending on the value of τ .) After that, a “feature difference” can be evaluated with $D_i = x_i^\tau(e_1) - x_i^\tau(e_2), i = 1, 2, \dots, N_D$. Based on all the D_i , the mean and variance of the feature difference can be estimated.

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

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

γ hypotheses ($\gamma = 50$ in the experiments) are generated and the one with the largest variance on feature difference will be found. Usually, a larger variance means that the data distribution spreads out more and the feature difference is more significant. Therefore the corresponding mean is used as the threshold to split the dataset. After this, one node will be built and the dataset will be partitioned into two parts. For each part, a new node will be further constructed in the same way. This process is repeated until the predefined maximum depth is reached.

Compared with other implementations of randomized visual vocabularies, e.g., LSH, the benefits of random indexing trees are numerous. In this chapter, let us point out four properties that are essential. First, each tree in the model is almost independent to others. Second, random indexing trees are fast to evaluate during the query stage.

The computation time only depends on the number of trees and the depth of each tree. Hence, it is usually faster than LSH based nearest neighbor search [9]. In the experiments, it can be seen that random indexing trees based weighting approach is over 300 times faster than LSH based approaches. This is of great importance if real-time action analysis is needed. Another advantage of random indexing trees compared with LSH is that, during the construction of each tree, data distribution of the STIPs is integrated, which means the tree construction is guided by the data density. This is one reason why random indexing trees has great speed gain but little performance loss. From that aspect, the structure of random indexing trees is flexible. Finally, by adding more trees to model, it can alleviate the affection of lacking query samples and well model the in-class variances. As shown in Fig. 5.2, for each vocabulary, only a small portion of nearest neighbors can be found. The benefits of multiple visual vocabularies are to increase the confidences of multiple matches and find as many nearest neighbors as possible.

5.3 Local Interest Point Matching

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

$$V^* = \max_{V \in \mathcal{D}} s(\mathcal{Q}, V), \quad (5.1)$$

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

Unlike previous single template action detection and retrieval [26], which can only take one positive sample for query, the proposed approach can integrate multiple

query samples and even negative ones. By introducing negative samples during the query phase, the algorithm is more discriminative. In addition, this approach enables interactive search by leveraging the labels obtained from user feedbacks.

Following the similar formulation in Eq. 3.1, the mutual information is used as the similarity function for $s(\mathcal{Q}, V)$. So we have:

$$\begin{aligned}
V^* &= \max_{V \subset \mathcal{D}} \mathcal{H}(\mathbf{C} = c_{\mathcal{Q}}, V) \\
&= \max_{V \subset \mathcal{D}} \log \frac{P(V|\mathbf{C}=c_{\mathcal{Q}})}{P(V)} \\
&= \max_{V \subset \mathcal{D}} \log \frac{\prod_{d_i \in V} P(d_i|\mathbf{C}=c_{\mathcal{Q}})}{\prod_{d_i \in V} P(d_i)} \\
&= \max_{V \subset \mathcal{D}} \sum_{d_i \in V} \log \frac{P(d_i|\mathbf{C}=c_{\mathcal{Q}})}{P(d_i)}.
\end{aligned} \tag{5.2}$$

Denote $s^{c_{\mathcal{Q}}}(d_i) = \log \frac{P(d_i|\mathbf{C}=c_{\mathcal{Q}})}{P(d_i)}$ as the mutual information between STIP d_i and query set $\mathcal{Q}$. In [9], $s^{c_{\mathcal{Q}}}(d_i)$ is computed based on one positive nearest neighbor point and one negative nearest neighbor point from d_i . However, nearest neighbor search in high dimensional space is very time consuming even with the advanced local sensitive hashing (LSH) technique [9]. Second, this approach is sensitive to noise, since only two points are used to compute its score. In order to address these problems, $s^{c_{\mathcal{Q}}}(d_i)$ is formulated as:

$$\begin{aligned}
s^{c_{\mathcal{Q}}}(d_i) &= \log \frac{P(d_i|\mathbf{C}=c_{\mathcal{Q}})}{P(d_i)} \\
&= \log \frac{P(d_i|\mathbf{C}=c_{\mathcal{Q}})P(\mathbf{C}=c_{\mathcal{Q}})}{P(d_i)P(\mathbf{C}=c_{\mathcal{Q}})} \\
&= \log \frac{P(\mathbf{C}=c_{\mathcal{Q}}|d_i)}{P(\mathbf{C}=c_{\mathcal{Q}})}.
\end{aligned} \tag{5.3}$$

In Eq. 5.3, $P(\mathbf{C} = c_{\mathcal{Q}})$ is the prior probability that can be computed as the ratio of the number of positive query STIPs to the total number of query STIPs.

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

Figure 5.2: An illustration of the indexing and action search based on randomized-visual-vocabulary.

Suppose N_T random indexing trees have been built offline from the database. At the query stage, all the STIP points in the query set $\mathcal{Q} = \mathcal{Q}_P \cup \mathcal{Q}_N$ (where $\mathcal{Q}_P$ and $\mathcal{Q}_N$ refer to positive query and negative query, respectively) are first extracted and distributed into the trees. Fig. 5.2 gives a two-dimension example where blue and black dot points represent the positive and negative STIPs, respectively. Each STIP point $d_i \in \mathcal{D}$ (red square in Fig. 5.2) falls into one of the leaves of a tree. Each leaf node contains several STIP points $d_q \in \mathcal{Q}$. In order to compute the posterior $P(\mathbf{C} = c_{\mathcal{Q}}|d_i)$, the information from all the leaves which contain d_i is integrated. Suppose d_i falls into a leaf with N_k^+ positive query STIP points and N_k^- negative points for tree T_k , then $P(\mathbf{C} = c_{\mathcal{Q}}|d_i)$ can be computed as:

$$P(\mathbf{C} = c_{\mathcal{Q}}|d_i) = \frac{1}{N_T} \sum_{k=1}^{N_T} \frac{N_k^+}{N_k^+ + N_k^-}. \quad (5.4)$$

As seen from Eq. 5.4, the voting strategy can integrate negative query samples, which makes the proposed algorithm more discriminative.

Eq. 5.3 can hence be rewritten as:

$$\begin{aligned} s^{c_{\mathcal{Q}}}(d_i) &= \log P(\mathbf{C} = c_{\mathcal{Q}}|d_i) - \log P(\mathbf{C} = c_{\mathcal{Q}}) \\ &= \log \frac{1}{N_T} \sum_{k=1}^{N_T} \frac{N_k^+}{N_k^+ + N_k^-} - \log P(\mathbf{C} = c_{\mathcal{Q}}). \end{aligned} \quad (5.5)$$

However, in the case where there are no negative query samples available ($\mathcal{Q}_N = \emptyset$),

Eq. 5.4 is slightly modified to:

$$P(\mathbf{C} = c_Q | d_i) = \frac{1}{N_T} \sum_{k=1}^{N_T} \frac{N_k^+}{M}, \quad (5.6)$$

where M is a normalization parameter. And Eq. 5.5 can be written as

$$\begin{aligned} s^{c_Q}(d_i) &= \log \frac{1}{N_T} \sum_{k=1}^{N_T} \frac{N_k^+}{M} - \log P(\mathbf{C} = c_Q) \\ &= \log \frac{1}{N_T} \sum_{k=1}^{N_T} N_k^+ - \log M - \log P(\mathbf{C} = c_Q). \end{aligned} \quad (5.7)$$

Further introduce a parameter $A = -\log M - \log P(\mathbf{C} = c_Q)$, which is set empirically in the experiments.

Each tree is a feature space partition as shown in Fig. 5.2. Usually, STIP points in the same leaf node are similar. The score evaluation (Eq. 5.5) on the trees can be explained intuitively by a “dying” process. Assume each positive query STIP point is blue and each negative point is black. For each query point, let us pass it down each tree. The leaf that the point falls in is dyed in the same color as the query point. Each leaf keeps a count of the number of times it is dyed by blue and a count of the number of times it is dyed by black after passing all the positive and negative query points down the trees. If a leaf’s blue count is larger than the black count, it is more likely to belong to the positive region, and vice versa. Given a point d_i (red square point in Fig. 5.2) in the dataset, to compute its score with respect to the positive queries, let us pass it down each tree. From each tree, let us find the leaf that d_i falls in. The blue counts and black counts of all the leafs in all the trees that d_i falls in are combined to estimate its posterior $P(\mathbf{C} = c_Q | d_i)$. The function of random indexing trees is like a special kernel, as shown by the yellow regions in Fig. 5.2. The idea of dying process is not only limited to trees but also applicable to any vocabulary structure.

Since random indexing trees share some similarity with random forest based work, e.g. [5], let us further compare the technique with random forest here. First, random

indexing trees are constructed in an unsupervised manner for class-independent video database indexing, while traditional random forests are constructed in a supervised manner. Second, in [5], random forest is used to vote for the hypothesized center positions through Hough voting, while random indexing trees do not rely on Hough voting. Thus, the non-trivial scale estimation of [5] is partially solved through the branch and bound search, which avoids an exhaustive search of all possible scales. Third, random indexing trees generate both positive and negative voting scores, thus it is more discriminative compared to [5], which generates only positive votes based on the frequency. Finally, the proposed algorithm uses random indexing trees for density estimation, which has been less exploited before. The detail discussion of random trees can be found in Section 2.6.

5.4 Efficient Action Search

5.4.1 Hierarchical Sub-volume Search

After computing the scores for all the STIP points in the database, the approach in [9] is followed to search for subvolumes in each video in the database. However, as stated in Chapter 3, there are two limitations in the subvolume search method proposed by [9]. First, multiple rounds of branch and bound search are needed to run if more than one instance are to be detected. In addition, the computational cost is extremely high when the video resolution is high.

In Chapter 3, three speeding up techniques have been proposed to reduce the computational cost of branch-and-bound search. They are spatial-downsampling, λ search and Top-K search. Although significant computational advantages have been achieved, it could still take 26mins to search one hour video database. This is unacceptable for

a on-line application. To further reduce the computational cost, a coarse-to-fine hierarchical search is proposed here, which employs the three proposed techniques in Chapter 3. The computational complexity of the proposed branch-and-bound search in Chapter 3 is $O(m^2n^2t) + O(Kmnt)$, where m, n, t are the width, height and duration of the database video and K refers to the top K results. Obvious, the most effective way to reduce the computational cost is to reduce the spatial resolution of the video. Thus, spatial-downsampling is performed to compress the search space. The following error bound for downsampling is proposed in Chapter 3.

$$f^s(\tilde{V}^*) \geq (1 - \frac{s * h + s * w + s^2}{wh})f(V^*), \quad (5.8)$$

where $\tilde{V}^* = \operatorname{argmax}_{V \in \mathcal{D}^s} f^s(V)$ denotes the optimal subvolume in the downsampled search space $\mathcal{D}^s$, $f(V) = \mathcal{H}(\mathcal{Q}, V)$, and V^* refers to the optimal subvolume in the original search space with width w and height h . With the help of this error bound, the top searched list can be relaxed to include more results for a further round of re-ranking. Suppose the top K results are needed from the database, the top- K search in Chapter 3 with spatial downsampling $2a$ is employed to obtain the first round results.

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

$$\frac{s * h + s * w + s^2}{wh} = 56.3\%. \quad (5.9)$$

So $\theta = 0.437f^s(\tilde{V}^{(K)})$ is chosen to filter the first round results. Then, for each remaining subvolume $\tilde{V}^{(k)}$ from the first round, the spatial size is extended with 30 pixels in each direction and another round of branch-and-bound search is performed. Different from previous round, which is running over all over the video space, this round of search is performed over the filtered 3D-subvolumes (extended with 30 pixels in each spatial direction). λ search Chapter 3 with $\lambda = f^s(\tilde{V}^{(K)})$ and downsampling factor a is used

in this round of search.

Despite only two rounds of search, random indexing trees based action detection algorithm outperforms the other algorithms. The proposed algorithm can be extended to more rounds of search. This is especially useful to handle high-resolution videos. As shown in Table 5.3, the efficient two-round branch-and-bound search only costs 24.1 seconds to search a database of one hour long 320×240 videos. This is over 60 times faster than the approach in Chapter 3 and 2900 times faster than [9]. Even for a 5-hour large database, it only costs 37.6 seconds to respond to the users. Besides from the speed advantages, in the experimental part, it can be found that the performance does not compromise. The top retrieved examples are with great accuracy by this coarse-to-fine search algorithm.

5.4.2 Refinement with Hough Voting

Although the search algorithm can successfully locate the retrieved actions, the localization step may not be accurate enough, as can be seen from the first row of Fig. 5.4. This motivates us to add a refinement step. The idea is to back-project the initial sub-volume into the query video. Based on the matches of STIPs, the action center (only in the spatial domain) can be voted. Fig. 5.3 is an illustration of the Hough refinement step.

Suppose the initial results have been obtained from the down-sampled branch and bound search (the blue region in the left image of Fig. 5.3). For all the STIP points within the detected subvolume, let us match them with the STIPs in the query video clip, either by trees or Nearest Neighbor search. Then the shift from the matched STIPs in the query will vote for the center of the retrieved action. To simplify the problem, let us only consider one fixed scale but smooth the votes with a Gaussian kernel. After

Figure 5.3: Illustration for Hough refinement.

considering all the votes, the center of the retrieved action is the position with the largest vote (the red cross in the third image of Fig. 5.3). To recover the spatial extent, the spatial scale of the action is set to be the smallest sub-volume which includes the initial retrieved region and the temporal scale is fixed to the initial retrieved result.

After the refinement, the blue region in the right image of Fig. 5.3 gives an illustration of the revised result compared with original round of result, i.e. the left image. Both quantitative results, as shown in Fig. 5.7, and empirical results in Fig. 5.4 show that the refinement step can successfully improve the retrieved results.

5.4.3 Interactive Search

The performance of action retrieval system is constrained by the limited number of queries. To show that the retrieval system can achieve better results when more queries are provided, an interaction step is added to facilitate human interaction. There are two major advantages of the interaction step. The first is to allow the user to express

what kind of action he/she wants to retrieve. Another advantage is that the proposed system can benefit from more query samples after each round of interaction.

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

The proposed algorithm for action search.

Require:

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

Ensure:

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

- 1: **Voting:** Based on Eq. 5.7, vote the database STIP points by the random indexing trees (Section 5.3).
- 2: **Searching:** Coarse round search with downsampling factor $2a$, discussed in Section 5.4.1, to get the top M retrieved results, where M depends on Eq. 5.8.
- 3: **Reranking:** Rerank the M sub-volumes from the first round of results with branch and bound search (downsampling factor a).
- 4: **Refinement:** [optional] Hough refinement discussed in Section 5.4.2.

Algorithm 4: The proposed algorithm for action search

5.4.4 Computational Complexity

For the action retrieval system, there are two major runtime costs: voting and searching. The computation complexity is $O(N_s T_d N_T)$ for the voting step where N_s refers to the number STIPs in a query clip, T_d refers to tree depth, and N_T refers to the number trees in a forest. As shown in the Table. 5.3, the voting time is negligible compared to

Figure 5.4: An illustration of the retrieval results (without and with Hough refinement). For each row, the first image represents one query frame and followed with seven highest ranked retrieved results. No user feedback is employed here.

the search time. For action search, the worst time complexity is

$$\begin{aligned}
T = & O((m/2a)^2(n/2a)^2t) + O(\hat{K}(m/2a)(n/2a)t) \\
& + O((\hat{m}/a)^2(\hat{n}/a)^2\hat{t}) + O(K(\hat{m}/a)(\hat{n}/a)\hat{t})
\end{aligned} \tag{5.10}$$

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

5.5 Experimental results

To validate the proposed algorithm, five experiments on four datasets have been discussed in this section. In Table. 5.1, the four datasets are listed for the validation of the search algorithms. Sample frames from the four datasets can be found in Fig. 5.5. In order to give a quantitative comparison with other work, a cross-dataset action de-

5.5.1 Action Detection

The random-indexing-trees strategy is validated with a challenging action detection experiment first. Since handwaving is one of the actions that have a lot of practical uses, handwaving action is to be detected in this experiment. The model is trained with KTH dataset (with 16 persons in the training part) and then perform experiments on a challenging dataset (MSR II) of 54 video sequences, where each video consists of several actions performed by different people in a crowded environment. Each video is approximately one minute long.

Figure 5.6: Handwaving detection on MSR II dataset based on precision-recall curve (AP means the average precision).

Fig. 5.6 compares the precision-recall curves for the following methods (the resolution for the original videos is 320×240):

- (i) ASTBB (Accelerated Spatio-Temporal Branch-and-Bound search) [8] in low resolution score volume (frame size 40×30),

Figure 5.7: Action search on MSR II dataset based on precision-recall curve.

- (ii) Multi-round branch-and-bound search [9] in low-resolution score volume (frame size 40×30),
- (iii) Top-K search in down-sampled score volume Chapter 3 (size 40×30),
- (iv) ASTBB [8] in 320×240 videos,
- (v) Random indexing trees based voting followed by Top-K search in down-sampled score volume (size 40×30).

The first four methods ((i)-(iv)) employ the LSH based voting strategy [9]. The measurement of precision and recall is the same as those described in [9]. To compute the precision, a true detection is considered 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, a hit is considered if: $\frac{\text{Volume}(V^* \cap G)}{\text{Volume}(V^*)} > \frac{1}{8}$. According to Fig. 5.6, random indexing trees based action detection outperforms the other algorithms. Compared with LSH voting strategy ((i)-(iv)), the voting score based on unsupervised random indexing trees is more discriminative and robust. The underlying reason is that random trees are data-sensitive, i.e. the data distribution is modeled when constructing the trees. Besides, since LSH only uses two nearest neighbors for

voting, the results are easily corrupted by noise. In Chapter 3, random forest is employed to do action detection. However, the difference is that trees in Chapter 3 are constructed in a supervised manner, which means that the label information is utilized when splitting the nodes, while random trees are unsupervised built with the purpose of modeling the underlying data distribution.

5.5.2 Action Retrieval on MSR II

To quantitatively evaluate the proposed action retrieval system, videos from MSR II are used as the database. The query samples are drawn from KTH dataset. As little work has been done before on MSR II dataset for action retrieval, to verify the system, let us compare the retrieval results with several action detection results from previous research. The evaluation is the same as that for action detection. For the implementations of random indexing trees, the number of trees in a forest is set as $N_T = 550$ and the maximum tree depth is set as 18. Fig. 5.7 compare the following three strategies on handwaving, handclapping and boxing actions (for the boxing action, each frame is flipped in the query video so that the boxing coming from both directions can be retrieved), respectively.

- (i) One positive query example without Hough refinement,
- (ii) One positive query example with Hough refinement,
- (iii) Cross-Dataset detection [14]¹,

As shown in Fig. 5.7, with a single query, the results ((i) and (ii)) are already comparable to (iii) for all three action types. This is quite encouraging because (iii) used

¹The STIP features in [14] are extracted in video resolution of 160×120 but 320×240 for other methods

all the training data while the proposed algorithm only uses a single query. Besides, Hough refinement scheme (ii) improves the results without Hough refinement (i).

Fig. 5.8 shows the experimental results of interactive action retrieval. The following six strategies (all of them are performed without Hough refinement) are compared.

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

In order to compare different visual vocabulary implementations, performance of LSH based indexing scheme is compared with that of using random indexing trees. The parameters for LSH are set to make the comparison fair.

It can be seen that when there is only one query example, random indexing trees based voting strategy is superior to LSH based voting strategy. When there are two query examples (one positive and one negative,) the retrieval results become worse than the one query case. The reason is that negative action type is hard to describe and a single example is usually not enough. However, the performance of the proposed system increases as more query samples are given. In particular, after two interaction steps, the retrieval results are better than the results obtained by other action detection systems ((i)-(iv) in Fig. 5.6), which utilize all the training data (256 examples).

Some illustrative results are provided in Fig. 5.4. For each query, seven subvolumes with the highest scores are listed in the figure. The retrieved subvolumes are marked by

colored rectangles. The rectangle with cyan background indicates a “correct” retrieval. As shown in the first row of Fig. 5.4, some of the cyan colored regions in the results are focused on a subregion of the action region. But this can be relieved with Hough refinement as indicated in the second row. In short, the action retrieval system can get very good results among the top retrieved subvolumes on various actions types.

Figure 5.8: Interactive action retrieval on MSR II dataset based on precision-recall curve.

5.5.3 Action Retrieval on CMU database

CMU database [11] is another widely used database for action analysis. Since the annotation of the actions includes the entire human rather than the action itself (as can be seen from Fig. 5.9, the results only mark the region where the action happens), only some illustrative examples on this dataset are presented here. The CMU database includes 48 videos of total duration around 20 minutes. The resolution for these videos are 160×120 . Handwaving and bending actions are retrieved from the database where

Figure 5.9: Retrieval results from CMU dataset. For each row, the first image represents the query frame and the following seven images refer to the highest ranked retrieved results.

the query video for handwaving is from KTH and the query video for bending is from Weizmann dataset [77].

Fig. 5.9 shows the search results. For each row, the first image is one frame from the query video and the following 7 images are from the top-7 retrieved segments, respectively. The cyan region shows the positive detection while the yellow region shows the negative detection. Compared with handwaving, bending is a non-periodic action, which is more challenging due to simple motion pattern and little number of query STIP points. From this experiment, it can be seen that the proposed algorithm can handle the two actions with a large in-class variances and clutter background, even in the low-resolution and highly compressed videos.

5.5.4 Action Retrieval on Youtube Video

In this experiment, the proposed algorithm is validated with a challenging tennis serve action search from a Youtube video², which is also a non-periodic action. More action searches from Youtube videos will be available from the project website. The length for the database video is around 280s, with several tennis serving actions performed

²<http://www.youtube.com/watch?v=inRRaudOf5g>

Figure 5.10: An action search example with tennis serve action. Both the query and database videos are downloaded from Youtube. The seven frames in the first row represent the frames in the query video while the images from second to sixth rows show the top-5 retrieved samples.

by different actors under different views. The query video is a 2 second segment cut from another Youtube video³. The experiment is very challenging due to the following aspects. First, there are different scenes and players compared with the query clip. Besides, the serving actions are recorded in several different views. Second, the videos contain not only the serving action but also other actions as well. The proposed algorithm needs to differentiate the serving action from other actions. Third, besides the large inner class variances, the video quality is also not very good due to the high compression rate. The blue marked regions (the reason to use blue color is that it

³<http://www.youtube.com/watch?v=NQcmYTIrqNI>

differentiates with the background color) from 2nd to 6th rows of Fig. 5.10 are the top 5 retrieved sub-volumes based on the query video from the first row. It can be seen that the proposed algorithm achieves promising results on these experiments.

5.5.5 Action Retrieval on Large-scale Database

To validate the proposed algorithms can handle large scale dataset, a large database is built with more than 200 videos. The database include videos from datasets MSRII [9], CMU [11], VIRAT [73], Hollywood [75] and some videos downloaded from Youtube. The total duration is around 5 hours.

Four different actions (handwaving, handclapping, boxing and ballet spinning) are tested in this large dataset. Each experiment is done with only one query video, without any post-processing, e.g. Hough refinement. The query videos for the first three actions (around 15s for each action) are collected from KTH while the query video for ballet spinning is downloaded from Youtube (around 5s). In Fig. 5.11, five samples retrieved results are selected from the top-40 detections (for ballet spin, only the top-10 results are retrieved since there do not have so many ballet spin actions in the database). The first row gives seven frames from the query video while the second to fifth rows show the four positive results where the retrieved subvolumes are marked with cyan. The sixth row shows one negative result where the retrieved subvolumes are marked with yellow. Similarly, Fig. 5.12, Fig. 5.13 and Fig. 5.14 shows the results of handclapping, boxing and ballet spin, respectively. Besides, in Fig. 5.15, the retrieved performance (precision versus the number of top samples retrieved) is provided for the large-scale database. Based on the illustrative results, it can be seen that the proposed algorithm can well handle the large scale changes, clutter background, partial occlusion and low visual quality.

Based on the quantitative results in Fig. 5.15, it can be found that the handwaving action retrieval works quite well for the action retrieval system. The performance is also fine for the ballet spinning and handclapping actions. For the boxing action, since the motion itself is not discriminative, it causes a lot of false positives. But considering that only one query video is used for retrieval and the database size is around 5 hours video, the general performance for the proposed system is encouraging.

Figure 5.11: Handwaving action retrieval in large dataset. The first row illustrates the seven frames from query action while the following five rows give the retrieved examples. The positive examples are marked by cyan and the negative example is marked by yellow.

Figure 5.12: Handclapping action retrieval in large dataset. The first row illustrates the seven frames from query action while the following five rows give the retrieved examples. The positive examples are marked by cyan and the negative example is marked by yellow.

5.5.6 Implementations

To implement action search, there are several issues which need to be taken care for both the indexing stage and query stage. For the indexing part, the number of trees needs to be determined. An experiment is performed to evaluate the relationship between the number of trees and average precision. The test environment is the same as that discussed in Section. 5.5.2. According to Fig. 5.16, the number of trees become stable from 300 for the handwaving and boxing action but from 500 for the handclapping action. Moreover, if only one tree (traditional BOW model) is used, the system cannot be able to find any positive detections. The reason is that only a very limit-

Figure 5.13: Boxing action retrieval in large dataset. The first row illustrates the seven frames from query action while the following five rows give the retrieved examples. The positive examples are marked by cyan and the negative example is marked by yellow.

ed number of database STIP points are matched to the query STIP points and these matched STIP points are isolated from each other. Usually only the STIP point itself forms a detection. Hence, the number of trees needs to be increased to introduce more matches. But as shown in the figure, the performance will be stable when sufficient number of trees are given.

Based on Fig. 5.16, the number of vocabularies (trees) is fixed as 550 in the experiment setting. Moreover, the depth for the trees is set as 18 for most of the experiments (For Youtube video dataset, it is set as 15 because the database is of small size). In the query stage, the downsampling factor of the branch-and-bound search (referring

Figure 5.14: Ballet spin action retrieval in large dataset. The first row illustrates the seven frames from query action while the following five rows give the retrieved examples. The positive examples are marked by cyan and the negative example is marked by yellow.

as *a*) is first set as 16 in the coarse round of search and then refined as 8 for another round of search.

5.5.7 Computational Cost

For the action retrieval system, there are two major runtime costs: voting and searching. The total computational cost for the proposed system is listed in Table 5.3. The test environment is as follows. one query video is used, which is approximately 20 seconds long. Two datasets are tested in the experiments. The first database is MSR II, which consists of 54 sequences with 320×240 resolution. The second dataset contains 5

Figure 5.15: Action retrieval results from 5-hour large dataset.

hours long videos (discussed in Section. 5.5.5). To set the parameter θ in Section 5.4.1, w and h are averaged among the top K results and obtain an error bound based on the estimated w and h . With this error bound, θ can be computed similarly as in Eq. 5.9. A PC with 2.6G CPU and 3G memory is utilized. The operating system is Windows 7. Table 5.2 compares the voting cost: the random indexing trees based vocabulary implementation is much more efficient compared with LSH based implementation. According to Table 5.2, random indexing trees are over 300 times faster than LSH based indexing but with even superior performance from Fig. 5.8. For the searching cost, as shown in Table 5.3, the coarse-to-fine subvolume search scheme only costs 24.1s for all 54 video clips in MSR II, while Top-K search in Chapter 3 takes 26mins. This is even 2800 times faster than the branch and bound search in [9]. For the 5-hour large dataset, it even only costs 37.6s to retrieve the top-7 results. From the statistics of Table 5.3, it can be found that the increase of database size (from 1 hour to 5 hours)

Figure 5.16: The evaluation of average precision based on the number of trees.

do not increase a lot of computational cost (from 26.4s to 37.6s). The reason is that the search consists of two rounds: coarse search and fine search. The fine search time is almost the same for the two datasets since only the similar number of candidates received from the coarse round search is considered. On the other hand, due to the high down-sampling factor in the coarse round search, the computational burden for large dataset is not that intensive.

The total computation time is independent of the duration of the query videos. This means, when there are more queries, the total computation time only grows linearly with the feature extraction time, which is around 30s for a 20s sequence. For a very large database, like Youtube, it has little impact to the voting cost since the voting cost mainly depends on the number of trees and the depth of each tree. In order to deal with the increasing search complexity, parallel computing can be utilized in the first step of branch and bound search since the search for different video clips are mutually independent. As the number of candidates for search in the second step of the branch

Method	Voting Time (ms)	One sequence (s)
LSH [9]	173.48 $\pm$ 423.71	173.48
random indexing trees	0.537 $\pm$ 0.14	0.537

Table 5.2: CPU time consumed by STIP voting in MSR II dataset which consists of 870,000 STIPs.

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

Table 5.3: Total computation time of the retrieval system.

and bound search only depends on the number of retrieved results required by the user, the database size has little impact on the runtime cost for the second step.

5.6 Concluding Remarks

In this chapter, an unsupervised action retrieval system is proposed which can efficiently and effectively locate the subvolumes similar to the query video. Random-indexing-trees based visual vocabularies are introduced and demonstrated to perform well for the database indexing. By increasing the number of vocabularies, the large in-class variance problem can be relieved despite only one query sample available. In addition, the mutual information based voting method has the unique property that it is very easy to leverage feedback from the user.

Moreover, a coarse-to-fine subvolume search scheme is proposed, which results in a dramatic speedup over the proposed spatio-temporal branch-and-bound method proposed in Chapter 3. Cross-dataset experiments demonstrate that the proposed method

is not only fast to search higher-resolution videos, but also robust to action variations, partial occlusions, and cluttered and dynamic backgrounds. Various challenging experiments are performed to verify the proposed action search system. Apart from the superior performance, the proposed system is fast for on-line applications, for example, an action search can be finished in 24s from a 1 hour database and in 37s from a 5 hour database, which is significantly faster than the algorithms in Chapter 3 and Chapter 4. However, thanks to the limited training examples, the performance obtained in this chapter is degraded compared with previous methods proposed in Chapter 3 and Chapter 4, which leverage more training examples.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, a systematic study has been presented on the challenging human action detection and recognition problem. More specifically, this thesis is summarized in the following two points.

Firstly, one part of this thesis is targeted at the human action detection problem with spatio-temporal localization in the video space. To obtain efficient and effective action localization, two potential solutions have been proposed. On one hand, an efficient 3D branch and bound search algorithm is presented in Chapter 3 to determine the spatio-temporal locations of human actions. Novel techniques like spatial down-sampling and Top-K search are also employed to reduce the computational cost. This algorithm can be further extended by a coarse-to-fine search model as in Chapter 5 with the ability to perform a large-scale action search. On the other hand, propagative Hough voting is presented in Chapter 4 to optimize both the search accuracy and computational cost. By utilizing the spatio-temporal configuration of local interest points,

the matching accuracy can be significantly improved. Also, rather than enumerating all the 3D sub-volumes in the video space, Hough voting is performed to generate a small set of candidate sub-volumes, which makes the proposed algorithm efficient for action localization.

Secondly, local interest point based representation has been employed throughout the thesis. One central problem for this video representation is how to provide effective and efficient local interest point matching. To address this problem, visual vocabularies have been utilized. Instead of generating or learning one set of vocabulary to represent the human actions, multiple sets of vocabularies can essentially relieve the problem of large intra-class variation. To implement the vocabularies, different tree variations have been presented. Supervised trees in Chapter 3 optimizes the classification error to obtain better recognition accuracy. Unsupervised random projection tree in Chapter 4 improves the matching accuracy by leveraging the underlying data distribution from both labeled and unlabeled video data. The goal of random indexing tree in Chapter 5 is to provide efficient local interest point matching.

To summarize, this thesis has identified and addressed the problems in human action detection for three different application scenarios. Firstly, Chapter 3 has focused on the traditional action detection problem and an efficient algorithm is presented based on fully-supervised model. Secondly, Chapter 4 has addressed a challenging action detection problem when the labeled training data is limited. “Semi-supervised” propagative Hough voting has been proposed to leverage both labeled and unlabeled video data for effective action localization. Finally, an extreme case of action detection, where only one query sample is provided to search the similar action instances, is discussed in Chapter 5 and an unsupervised action search system is presented which is significantly faster than the previous localization algorithms.

In general, several efficient and effective solutions have been developed with promis-

ing results on the benchmarked datasets. In addition, the novel techniques, e.g., coarse-to-fine branch-and-bound search, propagative Hough voting, visual vocabularies and so on, in this thesis should be beneficial for other visual recognition problem as well.

6.2 Future Work

In the future work, more effort will be focused on three aspects: action recognition with RGBD cameras, multi-class action detection in unconstrained videos, and action prediction in the unsegmented videos.

Compared with the RGB cameras used in this thesis, depth camera can naturally provide the accurate human skeleton as well as depth information for action recognition. Especially with the development of depth camera, like Kinect camera [146] and Prime Sense camera [147], action recognition in the living room environment with RGBD cameras becomes more and more feasible. A lot of potential applications can be developed if such a system is available. For example, when the user is watching TV programs in the living room, a pickup of remote action can be recognized and some recommendations can be instantly provided to the users based on the current programs. Also, when the users are recognized to make a phone call action, the volume of TV programs should be turned down automatically. Such applications based on action recognition will make the living room experience more intelligent.

Multi-class Action detection in unconstrained videos is another interesting application. The algorithms discussed in this thesis can be utilized for multi-class action detection but the time complexity linearly increases with the number of action classes. Thus, the computational cost of action detection on the dataset with large number of action categories, e.g., UCF101 [121], is intensive. In the future, multi-class action detection with constant time complexity will be highly demanded for large-scale video

datasets.

Different from action detection, action prediction needs to predict the human action even the action is still on-going. Previously, the current human action prediction research, including the work in Section 4.4.3, mainly focused on the segmented videos. In the future work, online action prediction on the unsegmented surveillance videos would be of great interest to the research community and industry.

Appendix:

Proof of Theorem 3.1

V^* denotes the optimal sub-volume in $\mathcal{V}$: $f(V^*) = \max_{V \in \mathcal{V}} f(V)$. Assume $V^* = [x_1, x_1 + w - 1] \times [y_1, y_1 + h - 1] \times [t_1, t_2]$ where w and h denote the width and height of V^* , respectively. Denote $|V|$ as the number of voxels in V . It can be obtained that there exists an s-aligned subvolume $\tilde{V} = [\tilde{x}_1, \tilde{x}_1 + \tilde{w} - 1] \times [\tilde{y}_1, \tilde{y}_1 + \tilde{h} - 1] \times [t_1, t_2]$ such that

$$|(V^* \setminus \tilde{V}) \cup (\tilde{V} \setminus V^*)| \leq (s * h + s * w + s^2)(t_2 - t_1). \quad (7.1)$$

So

$$\frac{|(V^* \setminus \tilde{V}) \cup (\tilde{V} \setminus V^*)|}{|V^*|} \leq \frac{s * h + s * w + s^2}{wh}. \quad (7.2)$$

Suppose the total score of a sub-volume is proportional to its size in average, then we have

$$\frac{f((V^* \setminus \tilde{V}) \cup (\tilde{V} \setminus V^*))}{f(V^*)} \leq \frac{s * h + s * w + s^2}{wh}. \quad (7.3)$$

Then

$$\frac{f(V^*) - f(\tilde{V})}{f(V^*)} \leq \frac{s * h + s * w + s^2}{wh}. \quad (7.4)$$

After a re-arrangement of the items, we have:

$$f(\tilde{V}) \geq (1 - \frac{s * h + s * w + s^2}{wh})f(V^*) \quad (7.5)$$

Proof of Theorem 4.1

A random variable $\mathbf{x}^{(i)}$ is introduced with Bernoulli distribution to indicate whether there is a vote for the position l^* or not in the i th match. Denote q as the match accuracy and therefore $p(\mathbf{x}^{(i)} = 1) = q$. Let us introduce another random variable with Bernoulli distribution $\mathbf{y}^{(i)}$ to indicate whether there is a vote for the background position l_j (where $l_j \neq l^*$) or not in the i th match. Suppose each background position has an equal probability to be voted, then $p(\mathbf{y}^{(i)} = 1) = \frac{1-q}{N_H}$.

Based on the discussion in Section 4.3.1, $S(V(l^*), \mathcal{R})$ is proportional to the number of matches which vote to position l^* , i.e., the number of correct match pairs between the testing interest points and training interest points $\mathcal{R}$. In total, we have N_M matches. Assume all the matches are independent. The score function turns to:

$$\begin{aligned} S(V(l^*), \mathcal{R}) &\propto \sum_{i=1}^{N_M} \mathbf{x}^{(i)} \\ S(V(l_j), \mathcal{R}) &\propto \sum_{i=1}^{N_M} \mathbf{y}^{(i)}. \end{aligned} \tag{7.6}$$

Obviously, $\{\mathbf{x}^{(i)} - \mathbf{y}^{(i)}\}_{i=1}^{N_M}$ are i.i.d. random variables. The expectation can be computed as $E(\mathbf{x}^{(i)} - \mathbf{y}^{(i)}) = q - \frac{1-q}{N_H}$ and the variance is:

$$\begin{aligned} &Var(\mathbf{x}^{(i)} - \mathbf{y}^{(i)}) \\ &= Var(\mathbf{x}^{(i)}) + Var(\mathbf{y}^{(i)}) - 2cov(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \\ &= q(1-q) + \frac{(1-q)(N_H-1+q)}{N_H^2} + \frac{2q(1-q)}{N_H}. \end{aligned}$$

For simplicity, define

$$\sigma_{\mathbf{xy}}^2 = Var(\mathbf{x}^{(i)} - \mathbf{y}^{(i)}). \tag{7.7}$$

When $N_T \rightarrow \infty$, we have $N_M \rightarrow \infty$. According to the central limit theorem, we have

$$\frac{1}{N_M} \sum_{i=1}^{N_M} (\mathbf{x}^{(i)} - \mathbf{y}^{(i)}) \sim \mathcal{N}(q - \frac{1-q}{N_H}, \frac{\sigma_{\mathbf{xy}}^2}{N_M}).$$

That is

$$\begin{aligned} & \lim_{N_M \rightarrow \infty} P\left(\frac{1}{N_M} \sum_{i=1}^{N_M} (\mathbf{x}^{(i)} - \mathbf{y}^{(i)}) > 0\right) \\ &= 1 - \Phi\left(\frac{-(q - \frac{1-q}{N_H})\sqrt{N_M}}{\sigma_{\mathbf{xy}}}\right) \end{aligned} \quad (7.8)$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{x^2}{2}} dx$. Thus, we have $S(V(l^*), \mathcal{R}) > S(V(l_j), \mathcal{R})$ with probability $1 - \Phi\left(\frac{-(q - \frac{1-q}{N_H})\sqrt{N_M}}{\sigma_{\mathbf{xy}}}\right)$ when $N_T \rightarrow \infty$.

Specifically, let us follow the above analysis but with the additional condition $q > \frac{1}{N_H+1}$ to show the asymptotic property. According to the weak law of large numbers, we have $\forall \epsilon > 0$,

$$\lim_{N_M \rightarrow \infty} P\left(\left|\frac{1}{N_M} \sum_{i=1}^{N_M} (\mathbf{x}^{(i)} - \mathbf{y}^{(i)}) - (q - \frac{1-q}{N_H})\right| < \epsilon\right) = 1. \quad (7.9)$$

Now let $\epsilon = \frac{q - \frac{1-q}{N_H}}{2} > 0$, we have

$$\lim_{N_M \rightarrow \infty} P\left(\frac{1}{N_M} \sum_{i=1}^{N_M} (\mathbf{x}^{(i)} - \mathbf{y}^{(i)}) > \frac{q - \frac{1-q}{N_H}}{2}\right) = 1. \quad (7.10)$$

As mentioned above, we have $q \geq \frac{1}{N_H+1}$. Thus,

$$\lim_{N_M \rightarrow \infty} P\left(\sum_{i=1}^{N_M} \mathbf{x}^{(i)} - \sum_{i=1}^{N_M} \mathbf{y}^{(i)} > 0\right) = 1. \quad (7.11)$$

So, we have $S(V(l^*), \mathcal{R}) > S(V(l_j), \mathcal{R})$ when $N_T \rightarrow \infty$.

Author's Publications

Book

1. Gang Yu, Junsong Yuan, and Zicheng Liu "Human Action Analysis with Randomized Trees", Springer Briefs in Signal Processing, In press.

Journal

1. Gang Yu, Junsong Yuan, and Zicheng Liu "Propagative Hough Voting for Human Activity Detection and Recognition", *Accepted by IEEE Trans. on Circuits and Systems for Video Technology*
2. Gang Yu, Junsong Yuan, and Zicheng Liu "Action Search by Example using Randomized Visual Vocabularies", *In IEEE Trans. on Image Processing, Vol.22, Issue 1, pp. 377-390, 2013*
3. Gang Yu, Norberto A., Junsong Yuan, Zicheng Liu "Fast Action Detection via Discriminative Random Forest Voting and Top-K Subvolume Search", *In IEEE Trans. on Multimedia, Vol.13, Issue 3, pp. 507-517, 2011*

Conference

1. Gang Yu and Junsong Yuan "Scalable Forest Hashing for Fast Similarity Search", *International Conference on Multimedia and Expo (ICME) 2014*
2. Gang Yu, Junsong Yuan, and Zicheng Liu "Propagative Hough Voting for Human Activity Recognition", *In Proc. European Conference on Computer Vision (ECCV) 2012*
3. Yuning Jiang, Junsong Yuan, Gang Yu "Randomized Spatial Partition for Scene Recognition", *In Proc. European Conference on Computer Vision (ECCV) 2012*
4. Gang Yu, Junsong Yuan, and Zicheng Liu "Predicting Human Activities using Spatio-Temporal Structure of Interest Points", *In ACM Multimedia Conference, 2012*
5. Gang Yu, Junsong Yuan, and Zicheng Liu "Unsupervised Random Forest Indexing for Fast Action Search", *In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011*
6. Gang Yu, Junsong Yuan, and Zicheng Liu "Real-time Human Action Search using Random Forest based Hough Voting", *In ACM Multimedia Conference, 2011*

Bibliography

- [1] I. Laptev, “On space-time interest points,” *International Journal of Computer Vision*, vol. 64, no. 2-3, pp. 107–123, 2005.
- [2] G. Brodal, A. Jørgensen, “A linear time algorithm for the k maximal sums problem,” in *Mathematical Foundations of Computer Science*, pp. 442–453, 2007.
- [3] C. Schuldt, I. Laptev, B. Caputo, “Recognizing human actions: a local svm approach,” in *International Conference on Pattern Recognition*, pp. 57-62, 2004.
- [4] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld, “Learning realistic human actions from movies,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.
- [5] J. Gall, V. Lempitsky, “Class-specific hough forests for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1022-1029, 2009.
- [6] K.K. Reddy, J. Liu, M. Shah, “Incremental action recognition using feature-tree,” in *International Conference on Computer Vision*, pp. 1010-1017, 2009.
- [7] J. Yuan, Z. Liu, Y. Wu, “Discriminative Video Pattern Search for Efficient Action Detection,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, In press.
- [8] J. Yuan, Z. Liu, Y. Wu, Z. Zhang, “Speeding up spatio-temporal sliding-window search for efficient event detection in crowded videos,” in *ACM Multimedia Workshop on Events in Multimedia*, pp. 1-8, 2009.
- [9] J. Yuan, Z. Liu, Y. Wu, “Discriminative subvolume search for efficient action detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2442-2449, 2009.
- [10] E. Shechtman, M. Irani, “Space-time behavior based correlation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 405-412, 2005.

-
- [11] Y. Ke, R. Sukthankar, M. Hebert, "Event detection in crowded videos," in *International Conference on Computer Vision*, pp. 1-8, 2007.
- [12] Y. Hu, L. Cao, F. Lv, S. Yan, Y. Gong, T. Huang, "Action detection in complex scenes with spatial and temporal ambiguities," in *International Conference on Computer Vision*, pp. 128-135, 2009.
- [13] A. Bobick, J.W. Davis, "The recognition of human movement using temporal templates," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257-267, 2001.
- [14] L. Cao, Z. Liu, T. Huang, "Cross-dataset Action Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1998-2005, 2010.
- [15] G. Yu, A. Norberto, J. Yuan, Z. Liu, "Fast Action Detection via Discriminative Random Forest Voting and Top-K Subvolume Search," in *IEEE Transaction on Multimedia*, Vol.13, Issue. 3, pp. 507-517, 2011.
- [16] M.D. Rodriguez, J. Ahmed, M. Shah, "Action mach a spatio-temporal maximum average correlation height filter for action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.
- [17] Y. Kong, Y. Jia, Y. Fu, "Learning Human Interaction by Interactive Phrases," in *European Conference on Computer Vision*, 2012.
- [18] E. Boyer, D. Weinland, R. Ronfard, "Free viewpoint action recognition using motion history volumes," *IEEE Conference on Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 207-229, 2006.
- [19] Y. Ke, R. Sukthankar, M. Hebert, "Efficient visual event detection using volumetric features," in *International Conference on Computer Vision*, pp. 166-173, 2005.
- [20] G. Yu, J. Yuan, Z. Liu, "Action Search by Example using Randomized Visual Vocabularies," in *IEEE Transaction on Image Processing*, Vol.22, Issue 1, pp. 377-390, 2013.
- [21] M. Yang, F. Lv, W. Xu, K. Yu, Y. Gong, "Human action detection by boosting efficient motion features," in *International Conference on Computer Vision workshop*, Kyoto, Japan, Sept.29 - Oct.2, pp. 1-8, 2009.
- [22] Z. Lin, Z. Jiang, L. Davis, "Recognizing actions by shape-motion prototype trees," in *International Conference on Computer Vision*, pp. 444-451, 2009.

-
- [23] H. Jiang, M.S Drew, Z.N. Li, “Action Detection in Cluttered Video with Successive Convex Matching,” in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.20, pp.50-64, 2010.
 - [24] E. Shechtman, M. Irani, “Space-time behavior-based correlation,” in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol.29, pp.2045-2056, 2007.
 - [25] A. Norbert, Z. Liu, J. Yuan, “Efficient search of Top-K video subvolumes for multi-instance action detection,” in *International Conference on Multimedia and Expo*, pp. 328-333, 2010.
 - [26] K.G. Derpanis, M. Sizintsev, K. Cannons, R.P. Wildes, “Efficient action spotting based on a spacetime oriented structure representation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1990-1997, 2010.
 - [27] C.H. Lampert, M.B. Blaschko, T. Hofmann, “Efficient Subwindow Search: A Branch and Bound Framework for Object Localization,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.31, pp. 2129-2142, 2009.
 - [28] C.H. Lampert, “Detecting Objects in Large Image Collections and Videos by Efficient Subimage Retrieval,” in *International Conference on Computer Vision*, pp. 987-994, 2009.
 - [29] L. Breiman, “Random forests,” in *Machine learning*, Vol.45, pp.5-32, 2001.
 - [30] A. Bosch, A. Zisserman, X. Munoz, “Image Classification Using Random Forests and Ferns,” in *International Conference on Computer Vision*, pp. 1-8, 2007.
 - [31] V. Lepetit, P. Laguerre, P. Fua, “Randomized trees for real-time keypoint recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 775-781, 2005.
 - [32] F. Schroff, A. Criminisi, A. Zisserman, “Object class segmentation using random forests,” in *British Machine Vision Conference*, pp. 1-8, 2008.
 - [33] P. Wang, G.D. Abowd, J.M. Rehg, “Quasi-periodic event analysis for social game retrieval,” in *International Conference on Computer Vision*, pp. 112-119, 2009.
 - [34] K. Prabhakar, S. Oh, P. Wang, G.D. Abowd, J.M. Rehg, “Temporal causality for the analysis of visual events,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1967-1974, 2010.
 - [35] J. Liu, J. Luo, M. Shah, “Recognizing realistic actions from videos ”in the wild”,,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1996-2003, 2009.

-
- [36] R. Messing, C. Pal, H. Kautz, “Activity recognition using the velocity histories of tracked keypoints,” in *International Conference on Computer Vision*, pp. 104-111, 2009.
- [37] L. Duan, D. Xu, I.W. Tsang, J. Luo, “Visual event recognition in videos by learning from web data,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1959-1966, 2010.
- [38] A. Kovashka, K. Grauman, “Learning a hierarchy of discriminative space-time neighborhood features for human action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2046-2053, 2010.
- [39] J. Niebles, C.W. Chen, F.F. Li, “Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification,” in *European Conference on Computer Vision*, pp. 392-405, 2010.
- [40] H.J. Seo and P. Milanfar, “Detection of human actions from a single example,” in *International Conference on Computer Vision*, pp. 1965-1970, 2009.
- [41] K. Mikolajczyk, H. Uemura, “Action recognition with motion-appearance vocabulary forest,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.
- [42] Y. Yacoob, M.J. Black, “Parameterized modeling and recognition of activities,” in *Computer Vision and Image Understanding*, Vol.73, pp. 232-247, 1999
- [43] D. Ramanan, D.A. Forsyth, “Automatic annotation of everyday movements,” in *Neural Information Processing Systems*, pp. 1-8, 2003
- [44] T.K. Kim, R. Cipolla, “Canonical correlation analysis of video volume tensors for action categorization and detection,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1415-1428, 2008.
- [45] L. Cao, Y.L. Tian, Z. Liu, B. Yao, Z. Zhang, T. Huang, “Action Detection Using Multiple Spatio-temporal Interest Point Features,” in *International Conference on Multimedia and Expo*, pp. 340-345, 2010.
- [46] H.J. Seo, P. Milanfar, “Action recognition from one example,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.33, pp. 867 - 882, 2010.
- [47] Z. Li, Y. Fu, S. Yan, T. Huang, “Real-Time Human Action Recognition by Luminance Field Trajectory Analysis,” in *ACM Conference on Multimedia*, pp. 671-676, 2008.

-
- [48] G. Zhu, M. Yang, K. Yu, W. Xu, Y. Gong, “Detecting video events based on action recognition in complex scenes using spatio-temporal descriptor,” in *ACM Conference on Multimedia*, pp.165-174, Oct.19-24, 2009.
- [49] M. Breitenbach, R. Nielsen, G.Z. Grudic, “Probabilistic Random Forests: Predicting Data Point Specific Misclassification Probabilities,” in *Technical Report CU-CS-954-03, the University of Colorado at Boulder*, pp. 1-8, 2003.
- [50] http://en.wikipedia.org/wiki/Pointwise_mutual_information
- [51] A. Yao, J. Gall, L. Van Gool, “A hough transform-based voting framework for action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2061-2068, 2010.
- [52] T.H. Yu, T.K. Kim, R. Cipolla, “Real-time action recognition by spatiotemporal semantic and structural forest,” in *British Machine Vision Conference*, pp. 1-12, 2010.
- [53] S. Mukherjee, S.K. Biswas, D.P. Mukherjee, “Recognizing Interaction Between Human Performers Using ‘Key Pose Doublet’,” in *ACM Conference on Multimedia*, 2011.
- [54] H. Wang, M.M. Ullah, A. Klaser, I. Laptev, C. Schmid, “Evaluation of local spatio-temporal features for action recognition,” in *British Machine Vision Conference*, pp. 1-11, 2009.
- [55] J. Bentley, “Programming pearls: algorithm design techniques,” in *Communications of the ACM*, Vol. 27, pp. 865–873, 1984.
- [56] D. Tran, J. Yuan, “Optimal Spatio-Temporal Path Discovery for Video Event Detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [57] G. Yu, J. Yuan, Z. Liu, “Unsupervised Random Forest Indexing for Fast Action Search,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [58] B. Leibe, A. Leonardis, B. Schiele, “Robust object detection with interleaved categorization and segmentation,” in *International Journal of Computer Vision*, Vol.77, pp. 259–289, 2008.
- [59] X. Zhou, L. Chen, “Monitoring near duplicates over video streams,” in *ACM Conference on Multimedia*, pp. 521–530, 2010.

-
- [60] L. Shang, L. Yang, F. Wang, K.P. Chan, X.S. Hua, “Real-time large scale near-duplicate web video retrieval,” in *ACM conference on Multimedia*, pp. 531–540, 2010.
- [61] J. Sun, X. Wu, S. Yan, L.F. Cheong, T.S. Chua, J. Li, “Hierarchical spatio-temporal context modeling for action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2004–2011, 2009.
- [62] H.K. Tan, C.W. Ngo, R. Hong, T.S. Chua, “Scalable detection of partial near-duplicate videos by visual-temporal consistency,” in *ACM Conference on Multimedia*, pp. 145–154, 2009.
- [63] L. Xie, A. Natsev, J. Tesic, “Dynamic Multimodal Fusion in Video Search,” in *International Conference on Multimedia and Expo*, pp. 1499–1502, 2007.
- [64] H. Ning, T.X. Han, D.B. Walther, M. Liu, T. Huang, “Hierarchical space-time model enabling efficient search for human actions,” in *IEEE Transaction on Circuits and Systems for Video Technology*, Vol. 19, pp. 808–820, 2009.
- [65] J. Collomosse, G. McNeill, Y. Qian, “Storyboard sketches for content based video retrieval,” in *International Conference on Computer Vision*, 2009.
- [66] Y. Aytar, M. Shah, J. Luo, “Utilizing semantic word similarity measures for video retrieval,” in *IEEE Conference On Computer Vision and Pattern Recognition*, 2008.
- [67] A. Gaidon, M. Marszalek, C. Schmid, “Mining visual actions from movies” in *British Machine Vision Conference*, 2009.
- [68] N. Ikizler-Cinbis, R.G. Cinbis, S. Sclaroff, “Learning actions from the web,” in *International Conference on Computer Vision*, 2009.
- [69] P. Wang, G.D. Abowd, J.M. Rehg, “Quasi-periodic event analysis for social game retrieval,” in *International Conference on Computer Vision*, 2009.
- [70] R. Ji, H. Yao, X. Sun, S. Liu, “Actor-independent action search using spatiotemporal vocabulary with appearance hashing,” in *Pattern Recognition*, Vol. 44, pp. 624–638, 2011.
- [71] I. Laptev, P. Prez, “Retrieving actions in movies,” in *International Conference on Computer Vision*, 2007.
- [72] J.C. Niebles, H. Wang, F.F. Li, “Unsupervised learning of human action categories using spatial-temporal words,” in *International Journal of Computer Vision*, Vol.3, pp.299–318, 2008.

-
- [73] S. Oh, A. Hoogs, A.G. Amitha, etc “A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [74] D.H. Ballard, “Generalizing the Hough Transform to Detect Arbitrary Shapes,” in *Pattern Recognition*, Vol.13, No.2, p.111-122, 1981.
- [75] M. Marszałek, I. Laptev, C. Schmid, “Actions in Context,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [76] T.-K. Kim, R. Cipolla, “Canonical correlation analysis of video volume tensors for action categorization and detection,” in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2009.
- [77] L. Gorelick, M. Blank, E. Shechtman, M. Irani, R. Basri, “Actions as Space-Time Shapes,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, pp. 2247-2253, 2007.
- [78] G. Willems, T. Tuytelaars, L.V. Gool, “An efficient dense and scale-invariant spatio-temporal interest point detector,” in *European Conference on Computer Vision*, 2008.
- [79] P. Dollar, V. Rabaud, G. Cottrell, S. Belongie, “Behavior recognition via sparse spatio-temporal features,” in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.
- [80] H. Wang, A. Klaser, C. Schmid, C.L. Liu, “Action recognition by dense trajectories,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [81] A. Klaser, M. Marszałek, “A spatio-temporal descriptor based on 3D-gradients,” in *British Machine Vision Conference*, 2008.
- [82] H. Wang, M.M. Ullah, A. Klaser, I. Laptev, C. Schmid, “Evaluation of local spatio-temporal features for action recognition,” in *British Machine Vision Conference*, 2009.
- [83] J. Liu, B. Kuipers, S. Savarese, “Recognizing human actions by attributes,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [84] S. Sadanand, J.J. Corso, “Action bank: A high-level representation of activity in video,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [85] G. Yu, J. Yuan, Z. Liu, “Predicting Human Activities using Spatio-Temporal Structure of Interest Points,” in *ACM Conference on Multimedia*, 2012.

-
- [86] S. Maji, J. Malik, “Object Detection Using a Max-Margin Hough Transform,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [87] K. Schindler, L. Van Gool, “Action snippets: How many frames does human action recognition require?,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.
- [88] Y. Xie, H. Chang, Z. Li, L. Liang, X. Chen, D. Zhao, “A Unified Framework for Locating and Recognizing Human Actions,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [89] L. Wang, Y. Qiao, X. Tang, “Motionlets: Mid-Level 3D Parts for Human Motion Recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
- [90] G. Yu, J. Yuan, Z. Liu, “Propagative Hough Voting for Human Activity Recognition,” in *European Conference on Computer Vision*, 2012.
- [91] M. Raptis, I. Kokkinos, S. Soatto, “Discovering Discriminative Action Parts from Mid-Level Video Representations,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [92] A. Jain, A. Gupta, M. Rodriguez, L. Davis, “Representing Videos using Mid-level Discriminative Patches,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
- [93] G. Zhu, M. Yang, K. Yu, W. Xu, “Detecting video events based on action recognition in complex scenes using spatio-temporal descriptor,” *ACM Conference on Multimedia*, 2009.
- [94] G. Yu, J. Yuan, Z. Liu, “Real-time Human Action Search using Random Forest based Hough Voting,” in *ACM Conference on Multimedia*, 2011.
- [95] H. Wang, C. Schmid, “Action Recognition with Improved Trajectories,” in *International Conference on Computer Vision*, 2013.
- [96] A. Efros, A.C. Berg, G. Mori, J. Malik, “Recognizing action at a distance,” in *International Conference on Computer Vision*, 2003.
- [97] D.L. Vail, M.M. Veloso, J.D. Lafferty, “Conditional random fields for activity recognition,” in *International joint conference on Autonomous agents and multi-agent systems*, 2007.

-
- [98] S.F. Wong, T.k. Kim, R. Cipolla, “Learning motion categories using both semantic and structural information,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [99] X. Wang, X. Ma, E. Grimson, “Unsupervised activity perception by hierarchical bayesian models,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [100] C.C. Loy, T. Xiang, S. Gong, “Modelling activity global temporal dependencies using time delayed probabilistic graphical model,” in *International Conference on Computer Vision*, 2009.
- [101] U. Gaur, Y. Zhu, B. Song, A.R. Chowdhury, “A string of feature graphs model for recognition of complex activities in natural videos,” in *International Conference on Computer Vision*, 2011.
- [102] W. Brendel, S. Todorovic, “Learning spatiotemporal graphs of human activities,” in *International Conference on Computer Vision*, 2011.
- [103] Z. Zhang, D. Tao, “Slow feature analysis for human action recognition,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.34, pp.436-450, 2012.
- [104] S. O’Hara, B.A. Draper, “Scalable action recognition with a subspace forest,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [105] C.C. Chen, J.K. Aggarwal, “Modeling human activities as speech,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [106] Q. Le, W.Y. Zou, S. Yeung, A. Ng, “Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [107] W. Choi, K. Shahid, S. Savarese, “Learning context for collective activity recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [108] M. Hoai, F. Torre, “Max-margin early event detectors,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [109] M.S. Ryoo, “Human activity prediction: Early recognition of ongoing activities from streaming videos,” in *International Conference on Computer Vision*, 2011.
- [110] C. Silpa-Anan, R. Hartley, “Optimised KD-trees for fast image descriptor matching,” *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

-
- [111] M. Muja, D.G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” *International Conference on Computer Vision Theory and Applications*, 2009.
- [112] A. Fu, P. Chan, Y.L. Cheung, Y.S. Moon, “Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances,” *International Journal on Very Large Data Bases*, 2000.
- [113] M. McCartin-lim, A. McGregor, R. Wang, “Approximate Principal Direction Trees,” *International conference on Machine Learning*, 2012.
- [114] S. Dasgupta, Y. Freund, “Random projection trees and low dimensional manifolds,” *ACM Symposium on Theory of Computing*, 2008.
- [115] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, “Real-Time Human Pose Recognition in Parts from a Single Depth Image,” *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [116] V. Lepetit, P. Fua, “Keypoint recognition using randomized trees,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28(9), pp. 1465C79, 2006.
- [117] L. Breiman, “Random Forests,” in *Machine Learning*, 2001.
- [118] C. Leistner, A. Saffari, J. Santner, H. Bischof, “Semi-Supervised Random Forests,” in *International Conference on Computer Vision*, 2009.
- [119] D. Weinland, E. Boyer, R. Ronfard, “Action Recognition from Arbitrary Views using 3D Exemplars,” in *International Conference on Computer Vision*, 2007.
- [120] K.K. Reddy, M. Shah, “Recognizing 50 Human Action Categories of Web Videos,” in *Machine Vision and Applications*, 2012.
- [121] K. Soomro, A.R. Zamir, M. Shah, “UCF101 : A Dataset of 101 Human Actions Classes From Videos in The Wild,” in *arxiv*, 2012.
- [122] A. Patron-perez, M. Marszalek, A. Zisserman, I. Reid, “High Five : Recognising human interactions in TV shows,” in *British Machine Vision Conference*, 2010.
- [123] H. Uehne, H. Jhuang, E. Garrote, T. Poggio, T. Serre, “HMDB: A large video database for human motion recognition,” in *International Conference on Computer Vision*, 2011.
- [124] M. Rohrbach, S. Amin, “A database for fine grained activity detection of cooking activities,” *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

-
- [125] F. Torre, J. Hodgins, A. Bargteil, A. Collado, X. Martin, J. Macey, P. Beltran, “Guide to the Carnegie Mellon University Multimodal Activity (CMU-MMAC) Database,” *Technical report*, 2008
- [126] R. Messing, C. Pal, H. Kautz, “Activity recognition using the velocity histories of tracked keypoints,” in *International Conference on Computer Vision*, 2009
- [127] F. Yuan, V. Prinet, J. Yuan, “Middle-Level Representation for Human Activities Recognition: the Role of Spatio-temporal Relationships,” in *European Conference on Computer Vision Workshop on Human Motion*, 2010.
- [128] P. Scovanner, S. Ali, M. Shah, “A 3-dimensional sift descriptor and its application to action recognition,” *ACM Conference on Multimedia*, 2007.
- [129] J. Gall, A. Yao, N. Razavi, L. Van Gool, V. Lempitsky, “Hough forests for object detection, tracking, and action recognition,” in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, pp. 2188–2202, 2011.
- [130] M.S. Ryoo, C. Chen, J. Aggarwal, “An overview of contest on semantic description of human activities,” in *Semantic Description of Human Activities*, 2010.
- [131] A. Gionis, P. Indyk, R. Motwani, “Similarity search in high dimensions via hashing,” in *International Conference on Very Large Data Bases*, pp. 518–529, 1999.
- [132] M.S. Ryoo, J.K. Aggarwal, “Spatio-Temporal Relationship Match: Video Structure Comparison for Recognition of Complex Human Activities,” in *International Conference on Computer Vision*, 2009.
- [133] M. R. Amer, S. Todorovic, “A Chains Model for Localizing Participants of Group Activities in Videos,” in *International Conference on Computer Vision*, 2011.
- [134] W. Brendel, S. Todorovic, “Learning Spatiotemporal Graphs of Human Activities,” in *International Conference on Computer Vision*, 2011.
- [135] N. Razavi, J. Gall, L. Van Gool, “Backprojection revisited: Scalable multi-view object detection and similarity metrics for detections,” in *European Conference on Computer Vision*, 2010.
- [136] B. Yao, L. Fei-Fei, “Modeling mutual context of object and human pose in human-object interaction activities,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

- [137] B. Leibe, A. Leonardis, B. Schiele, “Robust Object Detection with Interleaved Categorization and Segmentation,” in *International Journal of Computer Vision*, 77(1-3), 259-289, 2007.
- [138] F. Moosmann, E. Nowak, F. Jurie, “Randomized clustering forests for image classification,” in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol.30, pp.1632-1646, 2008.
- [139] A. Klaser, M. Marszalek, “A spatio-temporal descriptor based on 3D-gradients,” in *British Machine Vision Conference*, 2008.
- [140] O. Boiman, E. Shechtman, M. Irani, “In defense of nearest-neighbor based image classification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.
- [141] O. Barinova, V. Lempitsky, P. Kholi, “On detection of multiple object instances using hough transforms,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
- [142] Y. Song, Y.T. Zheng, S. Tang, X. Zhou, Y. Zhang, S. Lin, T.S. Chua, “Localized Multiple Kernel Learning for Realistic Human Action Recognition in Videos,” in *IEEE Transactions Circuits and Systems for Video Technology*, pp.1193-1202, Vol. 21, 2011.
- [143] N. Razavi, J. Gall, P. Kohli, L.V. Gool, “Latent Hough Transform for Object Detection,” in *European Conference on Computer Vision*, 2012.
- [144] M. Singh, A. Basu, M. Mandal, “Human activity recognition based on silhouette directionality,” in *IEEE Transactions on Circuits and Systems for Video Technology*, pp.1280-1292, 2008.
- [145] N. Dalal, B. Triggs, C. Schmid, “Human detection using oriented histograms of flow and appearance,” in *European Conference on Computer Vision*, 2006.
- [146] <http://www.xbox.com/en-sg/Kinect>
- [147] <http://www.primesense.com/>