LEARNING TRANSFERABLE DISTANCE FUNCTIONS FOR HUMAN ACTION RECOGNITION AND DETECTION

by

Weilong Yang

B.Eng., Southeast University, China, 2007

a Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Spring 2010

APPROVAL

Name:	Weilong Yang
Degree:	Master of Science
Title of Thesis:	Learning Transferable Distance Functions for Human Action Recognition and Detection
Examining Committee:	Dr. Richard Vaughan Chair
	Dr. Greg Mori, Senior Supervisor
	Dr. Oliver Schulte, Supervisor
	Dr. Ze-Nian Li, SFU Examiner
Date Approved:	April 7, 2010

Declaration of
Partial Copyright Licence

Abstract

In this thesis, we address an important topic in computer vision, human action recognition and detection. In particular, we focus on a special scenario where only a single clip is available for training for each action category. This is a very natural scenario in many real-world applications, such as video search and intelligent video surveillance. We present a transfer learning technique called transferable distance function learning and apply it in human action recognition and detection. This learning algorithm aims to extract generic knowledge from previous training sets, and apply this knowledge to videos of new actions without further learning. It is experimentally demonstrated that the proposed algorithm can improve the accuracy of single clip action recognition and detection. Based on the learned transferable distance function, we further propose a cascade structure which can significantly improve the efficiency of an action detection system.

Keywords: human action recognition, human action detection, transfer learning, distance function learning.

Acknowledgments

First, I would like to thank my advisor Dr. Greg Mori and my collaborator Dr. Yang Wang for their guidance on this thesis. Their encouragements and suggestions have helped me through many obstacles during this thesis. Greg and Yang are knowledgable, kind, and patient. The knowledge and skills I learned from them will continue to help me in my future research. Thanks to my thesis committee members, Dr. Oliver Schulte and Dr. Ze-Nian Li for their insightful comments on this thesis. Thanks to Dr. Richard Vaughan for chairing my thesis defense.

I also would like to thank all members in SFU Vision and Media Lab, especially Mohammad Norouzi, Mani Ranjbar and Tian Lan. They helped me a lot on this thesis. Thanks to all my friends in SFU, I will never forget the joyful days I spent with them.

Finally, thanks to my family. This thesis is impossible without their support and sacrifice.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

On-line video sharing services, e.g. YouTube (www.youtube.com), are becoming increasingly popular for users to find videos and share their own videos. Tens of thousands of videos are being uploaded to YouTube every hour. How to explore those enormous number of videos and return videos of interest for users is becoming a very important problem. Imagine that if you have a short video clip that contains a "dancing" action, as shown in Figure 1.1, can a search engine automatically return a list of similar videos from the repository, and accurately locate the spatial-temporal position of this specific dancing action? Traditional text-based search engines cannot perform this task very well. Because first you may not know the keyword associated with this action. Secondly, the text-based search engine only scans video title or tags, which might be irrelevant to the video content. However, computer vision techniques would help us solve this action-related video search problem. One of main goals of computer vision is to enable computer to have the ability to analyze and understand videos and images. In this thesis, we focus on one subfield of computer vision - human action recognition and detection, which is to enable computers to understand the action in videos.

Besides action-related video search, another important application of human action analysis is in automatic video surveillance, in particular automatic abnormality detection in surveillance videos. Traditional video surveillance systems only provide infrastructure to capture and store videos. Human operators are asked to perform the task of abnormality detection, which is labor-intensive and demanding. With the help of automatic human action recognition and detection, instead of human monitoring, a computer can analyze the content in surveillance videos and fire an alarm if an abnormal action is happening. For example, as shown in figure 1.2, the human action detection algorithm can automatically

Figure 1.1: Example application of human action recognition and detection: video search. The left image illustrates the reference video clip input by the user, and the right image illustrates the top-1 YouTube video returned by the search engine.

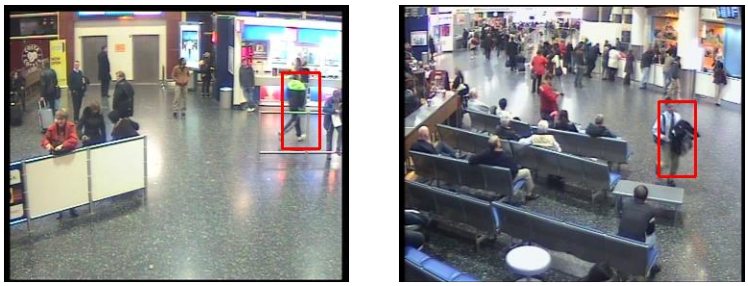


Figure 1.2: Example application of human action recognition and detection: automatic abnormality detection in surveillance videos. Human action detection algorithms can automatically locate (red bounding-box) the person who is running in an airport.

locate the person who is running in an airport.

Similar to object recognition and detection, human action recognition and detection can be formulated as a standard supervised learning problem that is to learn a classifier/detector from the training set. Many learning-based approaches have been proposed but they often highly rely on large training sets. However, in many real-world applications, it is unrealistic to assume that we have access to a large amount of training data. For example, in the video search application, we typically have only one short video clip submitted by the user for a particular action. In this thesis, we are interested in a special scenario where only a single short clip is provided for each action. We first introduce the background about human action recognition and detection in Section 1.1. In Section 1.2, we provide an overview of

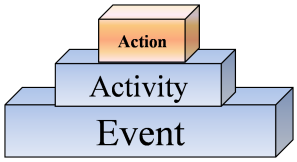


Figure 1.3: The terminology of human-related activity analysis in computer vision. Refer to text for more details.

how to learn a model with a single video clip. The contribution of this thesis is summarized in Section 1.3.

1.1 Human action recognition and detection

Human action recognition and detection is of great scientific interest in the computer vision community, and people use different terminologies to define human movements. First, we would like to clarify the terminology we are using. Human related activities can be generally categorized into "human action", "human activity", and "event", as shown in Figure 1.3. In this thesis, we refer "human action" to the atomic human movements, such as walking, running, hand-waving, etc. "Human activity" is used to refer to a series of atomic human actions. For example, the ball-kicking action performed by a soccer player can only be categorized as an "action" since it only consists of one atomic action - "kicking". However, one attack performed by several players would be categorized as an "activity" because it may consist of a series of actions, i.e., passing the ball, receiving the ball, and shooting. Whereas, the "event" is more complicated and it may consist of different activities. In this thesis, we limited ourselves on the analysis of human actions.

The goal of human action recognition is to classify a video sequence into one of several pre-defined categories based on the actions performed by the human in a video. In this thesis, we choose a *figure-centric* representation, as shown in Figure 1.4. Instead of directly running the action recognition algorithm on the original frame, we first apply a standard human detection and tracking algorithm, crop the frame, and then put the human figure in the center of the frame. We admit that the standard human detection and tracking algorithm may not achieve a perfect accuracy and thus decrease the action recognition performance. However, the use of this figure-centric representation would guarantee that we are recognizing the movements of human, rather than the translations caused by a moving

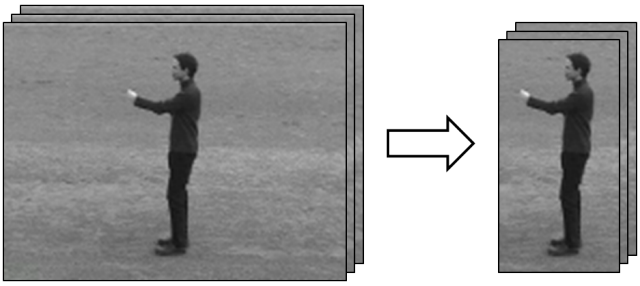


Figure 1.4: The figure-centric representation of a boxing video clip. Left image is an original boxing video clip and right image is the figure-centric representation of the boxing video clip.

camera.

The primary goal of action detection is distinct from that of action recognition - we would like to localize the spatial-temporal positions of the target action in a video, rather than getting a single class label for the entire video. We can achieve the goal of action detection by a human detection and tracking algorithm, followed by a modified action recognition algorithm. For example, we can first localize the human subjects from a video in both time and space using human detection and tracking, then apply a human action recognition algorithm to determinate if a localized human subject is performing the target action. However, in this thesis, we are also interested in developing an efficient human action detection algorithm which does not require any human detection and tracking pre-processing steps. One of the primary reasons is that most human detection and tracking algorithms are computationally inefficient.

Action recognition and detection are very challenging problems. One of main challenges is the action variation. Different people may perform the same action differently. Take the walking action as an example, different people may have different strides and different styles. Other challenges include viewpoint variation, illumination variation, and cluttered background. In order to be applied in many real-world scenarios, a good action recognition and detection algorithm must handle those problems well.

1.2 Learning with a single video clip

As mentioned earlier, our primary goal is to deal with the scenario where only one clip is available for a particular action. This scenario is of practical interests because for some

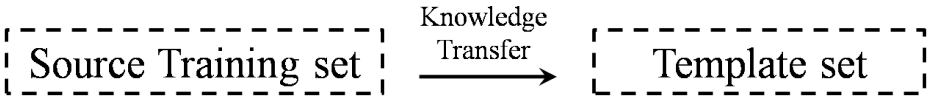


Figure 1.5: The flow of knowledge transfer, from source training set to template set.

specific human action like sports or dancing actions, for example the dancing action shown in Figure 1.1, it is costly to collect a very large training set. In particular, for the task of action-related video search, normally we only have one video clip provided by the user. Under this scenario, most supervised learning algorithms do not work well because of the lack of training data. One way to solve this single video clip problem in action recognition is to use the k-nearest neighbor algorithm, where $k = 1$. This is also called template matching in the computer vision literature. In order to obtain the action label of a test video, we can match it with the training video from each action category. This test video would be assigned the same class label as the training video that has the smallest matching distance with it.

Although we only have a single clip for some particular actions, on the other hand, for some simple actions such as walking and hand-waving, a large number of clips can be easily obtained from standard benchmark datasets, i.e. KTH [41] and Weizmann [6] datasets. The direct matching algorithm is able to achieve relatively good results, but we would still like to exploit the available labeled datasets to assist in action recognition from a single clip, even when the action of this clip is totally different from the actions in the labeled datasets. In machine learning, this is known as *transfer learning*. The goal is to leverage the knowledge from related tasks to aid in learning on a future task.

Following the terminology of transfer learning, we denote the fully labeled action data we already have at hand as the *source training set*. The single video clip from the action category we want to recognize is denoted as a *template action*, and the template actions from a pre-defined set of categories form the *template set*. Note that the class labels of actions in the source training set and the template set are different. In Chapter 3, we introduce the details of how to transfer the generic knowledge from the source training set to the template set. The flow of knowledge transfer is illustrated in Figure 1.5.

1.3 Contributions

The main contributions of this thesis involve the development of a parametrized distance function for comparing video clips. The distance function is defined as a weighted sum of distances between a densely sampled set of motion patches on frames of the video clips. This distance function is effective for action recognition in the impoverished training data setting. We further develop an algorithm for learning these weights, i.e. the distance function parameters. We develop a novel max margin-based *transfer learning* algorithm, inspired by the work of Frome et al. [21] and Ferencz et al. [19]. The learned weights are a function of patch features and can be generically transferred to a new action category without further learning. This learning method greatly improves the efficiency of our algorithm, and can improve recognition accuracy.

In this thesis, we also address the problem of efficient human action detection with only one template. The standard sliding-window approach is utilized to scan the template video against test videos, and the template video is represented by patch-based motion features. Using generic knowledge learned from previous training sets, we weight the patches on the template video, by a transferable distance function. Based on the patch weighting, we propose a cascade structure which can efficiently scan the template video over test videos. This action detection algorithm is evaluated on a human action dataset with cluttered background, and a ballet video with complex human actions. The proposed cascade structure not only achieves very reliable detection, but also can significantly improve the efficiency of patch-based human action detection, with an order of magnitude improvement in efficiency.

The work presented in this thesis has been published in [47] and [46]. Both of these papers are written in collaboration with Dr. Yang Wang and Dr. Greg Mori. The idea of transferable distance function learning is proposed and developed by myself, as well as most of the experimental work.

1.4 Outline

The rest of this thesis is organized as follows:

Chapter 2 reviews the related work in both machine learning and computer vision areas.

Chapter 3 focuses on human action recognition from a single clip per action. Detailed descriptions of four-channel motion feature and patch-based matching scheme we use are

provided. We propose the transferable distance function learning and evaluate the proposed method on three different human action datasets. This work has been published in [47].

Chapter 4 focuses on efficient action detection. We propose an efficient histogram-based four-channel motion feature. Based on the transferable distance function learning, we construct a cascade structure for the detection task. This work has been published in [46].

Chapter 5 concludes this thesis and discusses future work.

Chapter 2

Related work

In this chapter, we will review related work in both the machine learning and computer vision literature. Section 2.1 gives an overview of different learning scenarios related to transfer learning. It also gives a brief review of commonly used approaches in transfer learning. Section 2.2 reviews related work from the computer vision literature.

2.1 Related work in learning

Here we give a brief summary of transfer learning and other related learning problems that have been studied under various names (e.g. semi-supervised learning, self-taught learning, multi-task learning, domain adaptation, *etc.*). Interested readers are referred to [37] for a more detailed survey.

2.1.1 Related learning problems

The standard scenario of machine learning problems is *supervised learning*. We are given a set of labeled training data $\mathcal{D} = \{(x_n, y_n) \in \mathcal{X} \times \mathcal{Y} : 1 \leq n \leq N\}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ a finite label set. It is assumed that each (x_n, y_n) is drawn independently from a fixed, but unknown distribution p, i.e. $(x_n, y_n) \sim p(x, y)$. Our goal is to learn a function $f : \mathcal{X} \to \mathcal{Y}$ that can be used to predict the class label y for a new instance x. For example, if we want to learn to recognize images of "dogs" versus "cats", an instance x_n will be an image, and the label y_n is "dog" or "cat". Unfortunately, it is very expensive and time-consuming to collect a large number of labeled training instances. One possible

approach to deal with this issue is to use *semi-supervised learning* [50], which aims to learn a classification model from both labeled data $\{(x_i, y_i)\}_{i=1}^{N}$ and unlabeled data $\{x_j\}_{j=N+1}^{N+M}$, where $N << M$. In semi-supervised learning, the class labels and generative distributions of the unlabeled data are assumed to be the identical to those on the labeled data. In the "dog vs. cat" example, an unlabeled data instance x_j corresponds to an image of either *dog* or *cat*, but we simply do not know which label it is. A more challenging learning problem is *self-taught learning* [39], where the unlabeled data can have different class labels or generative distributions from the labeled data. Using the previous example, an unlabeled data instance in self-taught learning can be an image of anything, i.e. not limited to dogs or cats.

2.1.2 Transfer Learning

In our work, we are interested in *transfer learning*. Unlike previously mentioned learning scenarios, all the data in transfer learning are labeled. The goal of transfer learning to learn a predictive model by applying knowledge learned previously from other different but related task, i.e. it transfers knowledge from one supervised learning task to another. For example, if we want to learn to recognize "dogs" versus "cats" from two different kinds of labeled training datasets, called the source dataset $\mathcal{D}^{(s)} = \{(x_n^{(s)}, y_n^{(s)}) \in \mathcal{X} \times \mathcal{Y}^{(s)}\}$ and the target data $\mathcal{D}^{(t)} = \{(x_m^{(t)}, y_m^{(t)}) \in \mathcal{X} \times \mathcal{Y}^{(t)}\}$. The target dataset $\mathcal{D}_t$ contains instances labeled with either dogs or cats, i.e. $\mathcal{Y}^{(t)} = \{\text{dog}, \text{cat}\}$. The source dataset $\mathcal{D}^{(s)}$ contains instances with labeled with other categories (e.g. $\mathcal{Y}^{(s)} = \{\text{horse}, \text{tiger}, ...\}$). The goal of transfer learning is to exploit $\mathcal{D}^{(s)} \cup \mathcal{D}^{(t)}$ to build a model that recognize a new instance with an unknown label in $\mathcal{Y}^{(t)}$. Transfer learning is related to another learning problem called *domain adaptation* [23, 32, 8]. In domain adaptation, the source dataset and the target dataset have the same label set, i.e. $\mathcal{D}^{(s)} = \mathcal{D}^{(t)}$. But the source dataset and target dataset are drawn from two different distributions, i.e. $(x_n^{(s)}, y_n^{(s)}) \sim p^{(s)}(x, y)$, $(x_m^{(t)}, y_m^{(t)}) \sim p^{(t)}(x, y)$, and $p^{(s)} \neq p^{(t)}$.

Another closely related learning problem is *multi-task learning* [7]. The goal of multi-task learning is to learn different tasks together and assume there is some "relatedness" between different tasks. Ben-David and Schuller [5] provide a theoretical justification for multi-task learning. The problem setting of multi-task learning is almost identical to that of transfer learning. The only difference is that in multi-task learning, the goal is to learn a model that simultaneously do well on all the tasks. Using the previous example, the

multi-task learning might aim to learn to classify all the possible animals, not just "dogs" and "cats".

2.1.3 Commonly used approaches in transfer learning

Since transfer learning and multi-task learning are very closely related, most techniques developed for one of those two problems can be easily adapted to the other one. For ease of presentation, we will loosely call both problems "transfer learning" in the rest of this section.

An important assumption of transfer learning is that various tasks involved in the learning are somehow related. Otherwise it will be impossible to transfer the knowledge learned from one task to another one. Depending on the assumption of "relatedness", various techniques proposed in the literature roughly fall into two categories.

"Relatedness" via features: A lot of work in transfer learning assumes that different tasks are related by some intermediate feature representations. Argyrious et al. [2, 4, 3] learn a low-dimensional representation which is shared across multiple related tasks. If the intermediate representation shared across tasks is semantically informative, we can even perform zero-data [30] or zero-shot [36] learning, where the target task does not have any training data. This idea of transfering learned intermediate representation via related tasks can also be applied to solve regular classification problem (i.e. single task). For example, Ahmed et al. [1] demonstrate that classification can be improved by learning an intermediate feature representation from so-called "pseudo-tasks". Those "pseudo-tasks" are auxiliary tasks constructed to help with learning a good feature representation for the target task.

"Relatedness" via model parameters: Another popular approach in transfer learning is to assume that the model parameters associated in different tasks are related in some way. Let us denote the model parameters of the t-th task as $\mathbf{w}_t$ $(1 \leq t \leq T)$, where T is the number of tasks involved. Evgeniou et al. [12] assume $\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t$, where $\mathbf{w}_0$ are shared among all tasks and $\mathbf{v}_t$ are the parameters specific to the t-th task. In [48, 49], $\mathbf{w}_t$ $(1 \leq t \leq T)$ are related by assuming they are drawn from the same prior distribution $p(\mathbf{w})$, e.g. $p(\mathbf{w})$ can be a Gaussian process [48], or a t-process [49].

A limitation of the previous work in transfer learning is that the underlying classification model for each task is always assumed to be in a parametric form, e.g. a linear classification with parameters $\mathbf{w}$. It is not clear how to generalize those transfer learning techniques to other non-parametric classifiers, e.g. K-nearest neighbor (KNN). In this work, we develop

a technique for transferring "distance functions", which can be used in KNN classifiers. To the best of our knowledge, there has not been any previous work on transferring learning in this setting.

2.2 Related work in vision

2.2.1 Human action recognition and detection

A variety of action recognition and detection algorithms have been proposed and obtained high recognition accuracy on the standard KTH [41] and Weizmann [6] benchmark datasets. The vast majority of these methods use large amounts of training data, with either a leave-one-out (LOO) strategy or other splits of the data involving large amounts of training data for each action. The literature in this area is immense. We only give a brief review of the closely related work here.

Efros et al. [11] recognize the actions of small scale figures using features derived from blurred optical flow estimates. Fathi & Mori [16] learn an efficient classifier on top of these features using AdaBoost. Our method uses the same figure-centric representation, and defines patch distances using blurred optical flow. We learn a generic transferable distance function rather than individual classifiers, on smaller training sets.

A number of methods run interest point detectors over video sequences, and describe this sparse set of points using spatial and/or temporal gradient features[33, 10, 41]. In contrast with these methods, we use a densely sampled set of patches in our distance function. Our transfer learning algorithm places weights on these patches, which could be interpreted as a type of interest point operator, specifically tuned for recognition.

Shechtman and Irani [42] define a motion consistency designed to alleviate problems due to aperture effects. Distances between pairs of video clips are computed by exhaustively comparing patches centered around every space-time point. In our work we learn which patches are important for recognition, leading to a more efficient algorithm – though one could use motion consistency in place of blurred optical flow in a distance function.

Ke et al. [26, 27] define a shape and flow correlation based on matching of segmentations. Classification is done using a parts-based model [26] and an SVM trained on template distances in a LOO setting [27].

Jhuang et al. [24] describe a biologically plausible model containing alternating stages of spatio-temporal filter template matching and pooling operations. Schindler and Van Gool

[40] examine the issue of the length of video sequences needed to recognize actions. They build a model similar to Jhuang et al. [24] and show that *short* snippets can be effective for action recognition. Both of these methods use large splits for training data. Our work focuses instead on the amount of data needed, rather than the temporal length of the clips. Weinland and Ronfard [45] classify actions based on distances to a small set of discriminative prototypes selected in a LOO experiment.

Tran and Sorokin [43] propose a metric learning method for action recognition from small datasets. Our experiments use fewer frames (25 per training clip), and compare favourably in terms of accuracy.

2.2.2 Distance function learning

Our approach of learning transferable distance functions is inspired by the work of Frome et al. [22, 21]. They propose the learning of local distance functions and apply them in object recognition. Patch-based features are used to represent the object and also serve as the basis of the distance function. The distance between images i and j is defined as a weighted summation of *patch-to-image* distances as follows.

$$D_{ij} = \sum_{m=1}^{M} w_{i,m} d_{ij,m} = \langle \mathbf{w}_i \cdot \mathbf{d}_{ij} \rangle \tag{2.1}$$

where $w_{i,m}$ denotes the weight assigned to m-th patch on image i, and $d_{ij,m}$ is the *patch-to-image* distance which is the distance between m-th patch on image i and its best matched patch on image j. $w_{i,m}$ and $d_{ij,m}$ are also the m-th element in the vector $\mathbf{w}_i$ and $\mathbf{d}_{ij}$, respectively. The parameter $\mathbf{w}_i$ is learned by enforcing the constraint that the distance between similar images (from the same category) should be smaller than the distance between dissimilar images (from different categories).

Frome *et al.* [21, 22] propose to train a different distance function for every image in the training set. Two learning schemes are proposed, *focal learning* [21] and *global learning* [22]. In this thesis, the learning of the transferable distance function follows the framework of *focal learning*, so we only discuss the *focal learning* in this section. The focal learning is achieved by a similar-dissimilar triplet. Figure 2.1 gives an example of a triplet, where image i and j belong to the dog category and image k is in the face category. The center image i plays the role of *focal image*. The distance function with respect to the focal image is learned by enforcing that the distance from focal image to the other dog images is smaller than distance

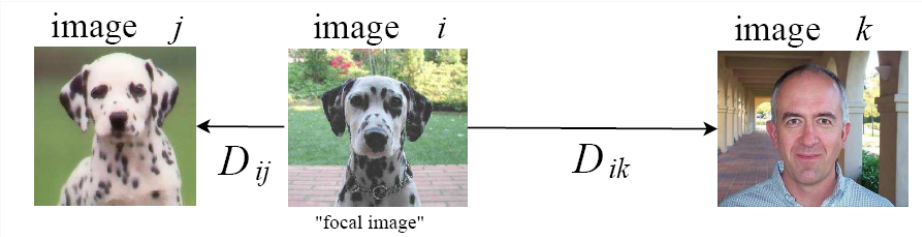


Figure 2.1: The illustration of focal learning configuration. Refer to text for more details. This image is from [20].

to the face image, that is $D_{ij} < D_{ik}$. It can be further expanded as $\langle \mathbf{w}_i \cdot \mathbf{d}_{ij} \rangle < \langle \mathbf{w}_i \cdot \mathbf{d}_{ik} \rangle$. The learning of parameter $\mathbf{w}_i$ is accomplished by the following max-margin formulation [21]:

$$\begin{aligned} \min_{\mathbf{w}_i, \xi_i} \quad & \frac{1}{2}\|\mathbf{w}_i\|^2 + C\sum_{jk}\xi_{ijk} \\ \text{s.t.} \quad & \langle \mathbf{w}_i \cdot (\mathbf{d}_{ik} - \mathbf{d}_{ij}) \rangle \geq 1 - \xi_{ijk}, \quad \forall i, j, k \\ & w_{i,m} \geq 0, \quad \forall m \\ & \xi_{ijk} \geq 0, \quad \forall j, k \end{aligned} \tag{2.2}$$

This formulation is similar to the primal formulation of SVM, except for the non-negative constraints $w_{i,m} \geq 0$, which is to avoid the problem of large *patch-to-image* distances implying a high similarity.

The intuition behind the distance function learning is that for a specific image, the visual features of this image, in particular the patch-based features, are not equally important for classification. Some patches are much more important than others. By learning the distance function, the algorithm would assign high weights to the important patches, and low weights or even zero weights to the less important patches. Note that the learning of distance functions cannot be directly applied to action recognition with single clip, since the focal learning process requires at least two clips for each action category.

2.2.3 Transfer learning in computer vision

In computer vision, the subarea of face/object identification has a long tradition of using ideas similar to transfer learning. In face identification, a system is trained from faces of thousands of people (source dataset). During the testing, a well-trained system can easily identify faces which do not exist in the source dataset. In particular, the work presented

in this thesis is inspired by the notion of "hyper-features", which is proposed by Ferencz *et al.* [19] for object identification. In a nutshell, hyper-features are properties of image patches that can be used to estimate the importance of those patches. These importance measurements can be used later in matching based object identification, and are transferable between different datasets. In our work, we use a similar idea to estimate the relative weights (i.e. importance) of motion patches extracted from video frames. We define the hyper-feature of a motion patch using a codebook representation. The main difference of our hyper-feature model with Ferencz et al. [19] is that our model is directly tied to the distance function used for the matching.

Other applications of transfer learning in computer vision include the one-shot learning of Fei-Fei et al. [17], in which object recognition models are built using priors learned from previously seen object classes. Farhadi et al. [15, 14] use comparative features for transferring distances between templates for sign language and multi-view action recognition. Quattoni et al. [38] perform transfer learning using kernel distances to unlabeled prototypes. Lampert et al. [28] learn to detect unseen object classes by considering object attributes as intermediate feature representation that can be transferred.

Chapter 3

Human Action Recognition

In this chapter, we consider the problem of human action recognition from a single clip per action. Each clip contains at most 25 frames. We work with a *figure-centric* representation in which a human detection and tracking algorithm has been run as a pre-processing step. A patch based motion descriptor and matching scheme have been proposed in Section 3.1, which can achieve promising results on three different action datasets with a single clip as the template. We also present a method for learning a transferable distance function for these patches in Section 3.2. The details of experiments and analysis are presented in Section 3.3.

3.1 Motion descriptors and matching scheme

We will classify the test video (we will call it *query video*) using the nearest neighbor (NN) classifier after computing the distances between the query video and each clip in the template set. The query video will be assigned to the action label according to the best matched template clip. The reason to use the NN classifier is that most other learning based approaches rely on complicated models with a large number of parameters, and thus cannot deal with the situation of very small training sets. In the following, we first introduce the motion descriptors used for representing a video clip (Section 3.1.1). Then we describe our patch-based matching scheme for comparing two video clips (Section 3.1.2).

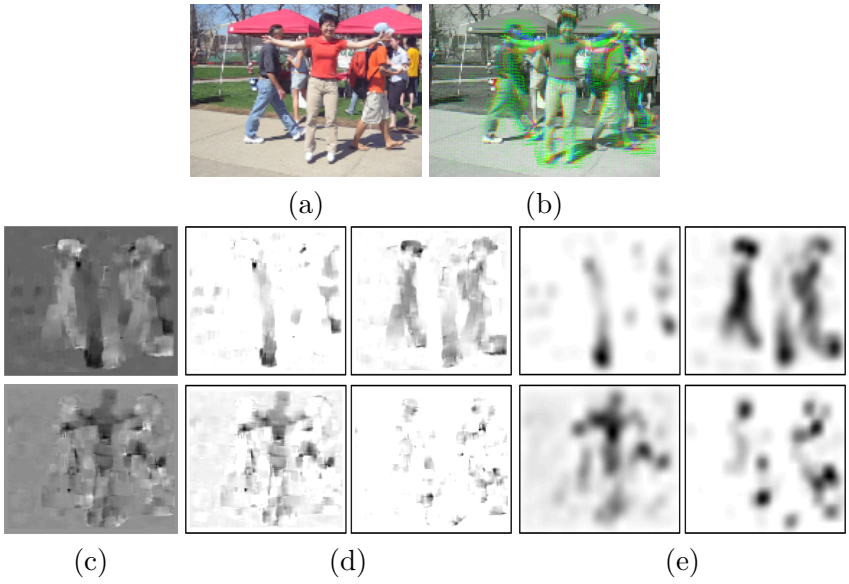


Figure 3.1: Construction of the motion descriptor. (a) original image; (b) optical flow; (c) x and y components of optical flow vectors F_x, F_y; (d) half-wave rectification of x and y components to obtain 4 separate channels $F_x^+, F_x^-, F_y^+, F_y^-$; (e) final blurry motion descriptors $Fb_x^+, Fb_x^-, Fb_y^+, Fb_y^-$.

3.1.1 Motion descriptors

In this work, we use a figure-centric representation of motion in which a standard human detector and tracking algorithm has been applied. The motion descriptors in Efros et al. [11] are used to represent the video frames. We first compute the optical flow at each frame. The optical flow vector field F is then split into two scalar fields, F_x and F_y corresponding to the x and y components of the flow vector. F_x and F_y are further half-wave rectified into four non-negative channels F_x^+, F_x^-, F_y^+, F_y^-, so that $F_x = F_x^+ - F_x^-$ and $F_y = F_y^+ - F_y^-$. Then, those four channels are blurred using a Gaussian kernel to obtain the final four channels Fb_x^+, Fb_x^-, Fb_y^+, Fb_y^- (see Figure 3.1).

3.1.2 Patch based action comparison

We compute the distance between two video clips by comparing the patches from both clips. Patch based methods are very popular in object recognition, due to the fact that local patches are more robust to pose variation than the whole object. We represent each patch using the four channel motion descriptor. Suppose the four channels for patch i are

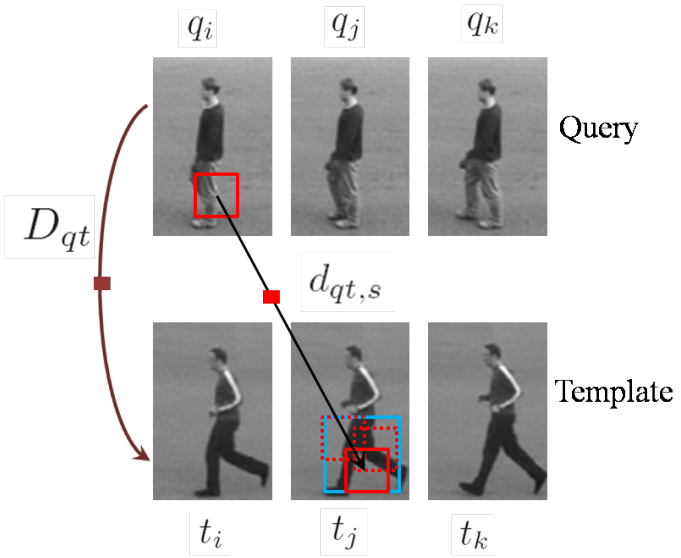


Figure 3.2: The comparison process between the query and template clips. $d_{qt,s}$ denotes the distance between the s-th patch on the query clip to its corresponding patch on the template clip. D_{qt} denotes the distance between query and template clips. The distance between clips is the sum of the distance from query frames to their best matched template frames. The frame-to-frame distance is the sum of the distance between best matching patches.

$\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, $\mathbf{a}_4$, and each channel has been concatenated to a vector. Similarly, the four channels for patch j are $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$, $\mathbf{b}_4$. We denote $\hat{\mathbf{a}_k} = [a_k^1 - \bar{a_k}, a_k^2 - \bar{a_k}, ..., a_k^n - \bar{a_k}]$, and $\hat{\mathbf{b}_k} = [b_k^1 - \bar{b_k}, b_k^2 - \bar{b_k}, ..., b_k^n - \bar{b_k}]$, where $\bar{a_k}$ and $\bar{b_k}$ are the mean values of channel $\mathbf{a}_k$ and $\mathbf{b}_k$ respectively, a_k^i denotes the i-th element in channel vector $\mathbf{a}_k$. The similarity between patch i and j is computed using normalized correlation, and the distance is given by

$$d(i,j) = C - \sum_{k=1}^{4} \frac{\hat{\mathbf{a}_k}^T \hat{\mathbf{b}_k} + \varepsilon}{\sqrt{(\hat{\mathbf{a}_k}^T \hat{\mathbf{a}_k} + \varepsilon)(\hat{\mathbf{b}_k}^T \hat{\mathbf{b}_k} + \varepsilon)}} \tag{3.1}$$

where C is a positive constant to make the distance non-negative, and ε is a small constant.

Different people may perform the same action differently. Take the walking action as an example, different people may have different strides, so the legs may appear in different positions of cropped frames. In order to alleviate the effect of such variations, we choose a local area search scheme. It is illustrated in Fig. 3.2. The distance between query and

template clips is:

$$D_{qt} = \sum_{i=1}^{M} \min_{j \in [1,N]} \left\{ \sum_{s=1}^{S} \min_{r \in R_s} d(q_{is}, t_{jr}) \right\} \tag{3.2}$$

where q_{is} denotes the s-th patch on the query frame i, and t_{jr} denotes the r-th patch on the template frame j. R_s is the corresponding search region of s-patch (the blue rectangle in Fig. 3.2). M and N are the frame numbers of query clip and template clip respectively. S is the total number of patches on the query frame.

In order to compute the clip-to-clip distance D_{qt} from query to template, we need to know the frame correspondence first. By considering temporal constraints, one can apply dynamic time warping(DTW) or other dynamic programming methods. In DTW, locality constraints are usually very necessary to compute the distance between two clips, especially when one clip is much longer than the other one, and then some DTW parameters will be introduced and require fine-tuning. However, in this work, for simplicity, we correspond each query frame to its closest neighbor among the template frames. This can result in several query frames corresponding to the same template frame. But it is reasonable since the query clip may contain repetitive actions and have variations in speed.

After obtaining the frame correspondence and local patch correspondence, D_{qt} is the sum of the elementary patch-to-patch distance as $D_{qt} = \sum_{s=1}^{M \times S} d_{qt,s}$, where $M \times S$ is the total number of patches on the query clip over space and time, $d_{qt,s}$ denotes the distance from the s-th patch on the query clip to its corresponding patch on the template clip.

In Section 3.3, we will show that even with such a simple motion descriptor and matching scheme, we can achieve very good results on three different datasets by only using one clip as template per action.

3.2 Learning a transferable distance function

3.2.1 Transferable distance function

The human visual system is amazingly good at learning transferable knowledge. For example, humans are adept at recognizing a person's face after seeing it only once. One explanation for this amazing ability is that people have learned to focus on discriminative features (e.g., eyes, nose, mouse) of a face, while not being distracted by other irrelevant features [19]. This idea of knowledge transfer has been exploited in the context of object

recognition and identification [19, 34, 17]. In particular, Ferencz et al. [19] propose to predict the patch importance for object identification by its visual feature called *hyper-feature*. The relationship between the hyper-feature and the patch importance is modeled using a generalized linear model.

Similarly, in human action recognition, we believe there exists a certain relationship between the importance and the appearance of a patch. For example, for a boxing action, the region around the punching-out arm is much more important than the still leg. In a hand-waving action, the arm parts are important too. Given a source training set, our goal is to learn the knowledge, such as "stretched-arm-like" or "bent-leg-like" patches are more likely to be important for action recognition. This knowledge will be "transferable" to unknown actions in the template and query datasets, since the algorithm will look for these patches and assign them high weights for the matching based recognition.

Inspired by work on learning distance function [21], we formulate our problem of learning the relationship into the framework of max-margin learning of distance functions. But the goal of our learning problem is different from that of Frome et al. [21]. The output of Frome et al. [21] is the weight associated with each image patch in the training data. In our problem, although we do get the weight as a by-product, we are more interested in learning the relationship between the patch appearance and its importance.

We define the hyper-feature of the i-th patch as $\mathbf{f}_i$, the weight assigned to this patch as w_i. The construction of the hyper-feature will be discussed in Section 3.2.4. We assume that $\mathbf{f}_i$ and w_i have the following relationship via the parameter $\mathbf{P}$:

$$w_i = \langle \mathbf{P} \cdot \mathbf{f}_i \rangle \tag{3.3}$$

Then we will have $\mathbf{w} = \mathbf{P}^T\mathbf{F}$, where each column of $\mathbf{F}$ refers to the hyper-feature vector of a patch, $\mathbf{w}$ denotes the vector which is the concatenation of the weights w_i. The process for the computing of $\mathbf{w}$ is illustrated in Figure 3.3.

Our goal is to learn $\mathbf{P}$ from the source training set. Then given any new action video, even if its action does not exist in the source training set, we will be able to compute the weight (i.e. importance) of each patch in the new video by Eqn. 3.3. In our work, we would like to estimate the importance of patches in the query video.

Combined with the learned distance function, the final clip-to-clip distance D_{qt} is defined

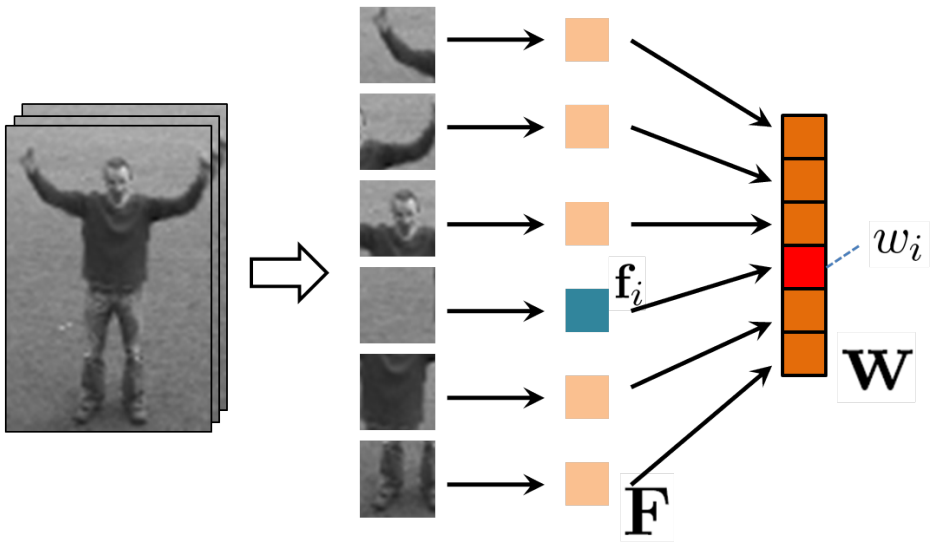


Figure 3.3: The process for computing the importance weights $\mathbf{w}$. For the patch-based action comparison, we first break a video clip into several patches. For patch i, we first compute its hyper-feature $\mathbf{f}_i$. Then, its associated importance weight w_i is computed by Eqn. 3.3.

as a weighted sum of all the elementary distances

$$D_{qt} = \sum_{s=1}^{S} w_{q,s} d_{qt,s} = \langle \mathbf{w}_q \cdot \mathbf{d}_{qt} \rangle \tag{3.4}$$

where $\mathbf{d}_{qt}$ is the distance vector, and each element denotes the elementary patch-to-patch distance $d_{qt,s}$. Note $w_{q,s}$ is the weight of the s-th patch on the query clip.

In Eqn. 3.3, we assume a linear relationship between hyper-feature and patch weight. In Eqn. 3.4, we also choose a linear relationship between elementary distance and the final clip-to-clip distance. For both equations, we believe nonlinear functions may work well. However, the nonlinearity will likely greatly increase the complexity of our learning process.

3.2.2 Max-margin formulation

The learning of $\mathbf{P}$ follows the focal learning framework in [21]. The distance function obtained by $\mathbf{w} = \mathbf{P}^T\mathbf{F}$ will satisfy the constraint that the distance between similar actions is smaller than dissimilar actions by the margin 1, that is

$$\begin{aligned} &\langle \mathbf{w}_i \cdot (\mathbf{d}_{ij} - \mathbf{d}_{ik}) \rangle > 1 \\ &\langle \mathbf{P}^T\mathbf{F}_i \cdot (\mathbf{d}_{ij} - \mathbf{d}_{ik}) \rangle > 1 \end{aligned} \tag{3.5}$$

where $\mathbf{d}_{ik}$ is the distance vector between the similar action i and k, and $\mathbf{d}_{ij}$ is the distance vector between the dissimilar action i and j. To avoid the problem of large patch-to-patch distances implying a high similarity, we enforce the non-negativity of the weights, $\langle \mathbf{P} \cdot \mathbf{f}_m \rangle \geq 0$. For simplicity, we replace $\mathbf{d}_{ij} - \mathbf{d}_{ik}$ as $\mathbf{x}_{ijk}$.

The max-margin optimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{P},\xi} \quad & \frac{1}{2}\|\mathbf{P}\|^2 + C\sum_{ijk}\xi_{ijk} \\ \text{s.t.} \quad & \langle \mathbf{P}^T\mathbf{F}_i \cdot \mathbf{x}_{ijk} \rangle \geq 1 - \xi_{ijk}, \quad \forall i,j,k \\ & \langle \mathbf{P} \cdot \mathbf{f}_m \rangle \geq 0, \quad \forall m \\ & \xi_{ijk} \geq 0, \quad \forall i,j,k \end{aligned} \tag{3.6}$$

where ξ_{ijk} is the slack variable and C is the trade-off parameter, similar to those in SVM. The hyper-feature $\mathbf{F}_i$ is known so we can write $\mathbf{Y}_{ijk} = \mathbf{F}_i \cdot \mathbf{x}_{ijk}$. The first constraint can be re-written as $\langle \mathbf{P} \cdot \mathbf{Y}_{ijk} \rangle \geq 1 - \xi_{ijk}$.

If we remove the second constraint, the optimization problem in Eqn. 3.6 will be similar to the primal problem of the standard SVM. The optimization problem is very similar to the one in Frome's work [21], but differs in the second constraint. Instead of the simple non-negative constraint $\mathbf{P} \geq \mathbf{0}$, like the one in [21], our constraints involve linear functions of the hyper-feature vectors.

The Lagrangian formulation of this optimization problem is:

$$\begin{aligned} \mathcal{L} \quad = \quad & \frac{1}{2}\|\mathbf{P}\|^2 + C\sum_{ijk}\xi_{ijk} - \sum_{ijk}\alpha_{ijk}[\langle \mathbf{P} \cdot \mathbf{Y}_{ijk} \rangle - 1 + \xi_{ijk}] \\ & - \sum_{ijk}\lambda_{ijk}\xi_{ijk} - \sum_{m}\mu_m\langle \mathbf{P} \cdot \mathbf{f}_m \rangle \end{aligned}$$

We can gather all the dual variables

$$\mathcal{L} \quad = \quad \frac{1}{2}\|\mathbf{P}\|^2 + \sum_{ijk}\alpha_{ijk}[\langle \mathbf{P} \cdot \mathbf{Y}_{ijk} \rangle - 1] + \sum_{ijk}\xi_{ijk}[C - \alpha_{ijk} - \lambda_{ijk}] - \sum_{m}\mu_m\langle \mathbf{P} \cdot \mathbf{f}_m \rangle$$

Since Lagrangian is linear with ξ_{ijk}, either ξ_{ijk} or $C - \alpha_{ijk} - \lambda_{ijk}$ must be zeros. So, we can remove ξ_{ijk} from Lagrangian and obtain one constraint as:

$$0 \leq \alpha_{ijk} \leq C \tag{3.7}$$

By taking the derivative of the remaining part of Lagrangian with respect to $\mathbf{P}$ and setting to zeros, we can get

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{P}} &= \mathbf{P} - \sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} - \sum_{m} \mu_m \mathbf{f}_m = 0 \\ \Longrightarrow \mathbf{P} &= \sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} + \sum_{m} \mu_m \mathbf{f}_m. \end{aligned} \tag{3.8}$$

Substituting Eqn. 3.8 back to the Lagrangian we can get:

$$\begin{aligned} & \Theta(\alpha, \mu) \\ = \; & \frac{1}{2}\|\sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} + \sum_{m} \mu_m \mathbf{f}_m\|^2 - \sum_{ijk} \alpha_{ijk}[\langle(\sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} + \sum_{m} \mu_m \mathbf{f}_m)\cdot \mathbf{Y}_{ijk}\rangle - 1] \\ & - \sum_{m} \mu_m[\langle(\sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} + \sum_{m} \mu_m \mathbf{f}_m)\cdot \mathbf{f}_m\rangle] \\ = \; & \frac{1}{2}\|\sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} + \sum_{m} \mu_m \mathbf{f}_m\|^2 - \|\sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk}\|^2 - \|\sum_{m} \mu_m \mathbf{f}_m\|^2 \\ & -2\langle\sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} \cdot \sum_{m} \mu_m \mathbf{f}_m\rangle + \sum_{ijk} \alpha_{ijk} \\ = \; & -\frac{1}{2}\|\sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} + \sum_{m} \mu_m \mathbf{f}_m\|^2 + \sum_{ijk} \alpha_{ijk} \end{aligned}$$

Then, the dual problem of Eqn. 3.6 can be written as follows

$$\begin{aligned} \max_{\alpha,\mu} \quad & -\frac{1}{2}\|\sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} + \sum_{m} \mu_m \mathbf{f}_m\|^2 + \sum_{ijk} \alpha_{ijk} \\ \text{s.t.} \quad & 0 \le \alpha_{ijk} \le C, \quad \forall i, j, k \\ & \mu_m \ge 0, \quad \forall m \end{aligned} \tag{3.9}$$

where the α_{ijk} and μ_m are the dual variables corresponding to the first and second constraints in Eqn. 3.6 respectively. The primal variable $\mathbf{P}$ can be obtained from the dual variables by Eqn. 3.8.

$$\mathbf{P} = \sum_{ijk} \alpha_{ijk}\mathbf{Y}_{ijk} + \sum_{m} \mu_m \mathbf{f}_m. \tag{3.10}$$

3.2.3 Solving the dual

Similar to [21], we solve the dual problem by iteratively performing updating on two dual variables. By taking the derivative of the dual with respect to one of the dual variables α_{abc} and then setting it to zero,

$$\frac{\partial \Theta}{\partial \alpha_{abc}} = \langle (-\sum_{ijk} \alpha_{ijk} \mathbf{Y}_{ijk} - \sum_{m} \mu_m \mathbf{f}_m) \cdot \mathbf{Y}_{abc} + 1 \rangle$$
$$= -\sum_{ijk} \alpha_{ijk} \langle \mathbf{Y}_{ijk} \cdot \mathbf{Y}_{abc} \rangle - \langle \sum_{m} \mu_m \mathbf{f}_m \cdot \mathbf{Y}_{abc} \rangle + 1$$
$$= -\sum_{ijk \neq abc} \alpha_{ijk} \langle \mathbf{Y}_{ijk} \cdot \mathbf{Y}_{abc} \rangle - \alpha_{abc} \|\mathbf{Y}_{abc}\|^2 - \langle \sum_{m} \mu_m \mathbf{f}_m \cdot \mathbf{Y}_{abc} \rangle + 1 \quad (3.11)$$

After setting Eqn. 3.11 to zero, we can obtain the updating rule for the dual variable α_{abc}. Similarly, we can get the updating rule for the dual variable μ_a. The two updating rules are as follows:

$$\alpha_{abc} \leftarrow \frac{1 - \sum_{ijk \neq abc} \alpha_{ijk} \langle \mathbf{Y}_{ijk} \cdot \mathbf{Y}_{abc} \rangle - \sum_m \mu_m \langle \mathbf{f}_m \cdot \mathbf{Y}_{abc} \rangle}{\|\mathbf{Y}_{abc}\|^2} \quad (3.12)$$

$$\mu_a \leftarrow \frac{-\sum_{ijk} \alpha_{ijk} \langle \mathbf{Y}_{ijk} \cdot \mathbf{f}_a \rangle - \sum_{m \neq a} \mu_m \langle \mathbf{f}_m \cdot \mathbf{f}_a \rangle}{\|\mathbf{f}_a\|^2} \quad (3.13)$$

After each round of update, we can simply clip the dual variables to their feasible regions. α_{abc} will be clipped to 0 if negative and to C if larger than C. μ_m will be clipped to zero if negative. See [22] for more details. After solving this dual problem, we can obtain $\mathbf{P}$ through Eqn. 3.10.

3.2.4 Hyper-features

Inspired by codebook approaches in object and scene categorization, we represent the hyper-feature of each patch as a $|V|$-dimensional vector $\mathbf{f}$, where $|V|$ is the codebook size. The i-th element of $\mathbf{f}$ is set according to the distance between the feature vector of this patch and the i-th visual word. The feature vector of each patch consists of histogram of oriented gradient (HOG) [9] and patch positions in the form of $h = \{g, x, y\}$, where g denotes the HOG descriptor of the patch. x and y are the coordinates of the patch in the frame. To construct the codebook vocabulary, we randomly select a large number of patches from the source training set, then run k-means clustering. The center of each cluster is defined as a codeword. The hyper-feature $\mathbf{f}_m$ for the m-th patch is constructed as follows

$$\mathbf{f}_m(v_i) = \frac{K_\sigma(D(v_i, h_m))}{\sum_{j=1}^{|V|} K_\sigma(D(v_j, h_m))} \quad (3.14)$$

where $\mathbf{f}_m(v_i)$ denotes the i-th element in the hyper-feature vector $\mathbf{f}_m$. $D(v_i, h_m)$ denotes the Euclidean distance between the i-th codeword and the patch m. K_σ is the Gaussian-shape

kernel as $K_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{x^2}{2\sigma^2})$. Note that Eqn. 3.14 leads to a generalized linear patch weighting model using Gaussian radial basis functions.

3.3 Experiments

We test our algorithms on three different datasets: KTH human action dataset [41], Weizmann human action dataset [6], and the cluttered action dataset [26]. We first give a brief overview of these three datasets, then present the experimental results.

3.3.1 Datasets

KTH dataset: The KTH human action dataset contains six types of human actions (boxing, hand-waving, hand-clapping, jogging, running and walking) performed several times by 25 subjects in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes and indoors. In total, there are 599 videos. Following the original setup, each video is divided into four sequences. After computing the motion descriptor, we run the human detection and tracking using the code provided by Felzenszwalb et al. [18]. All the frames and motion descriptors have been cropped to 90×60 and the human figure is put in the center of the frame.

On this dataset, the performance is saturating, with results from $90\% - 94\%$ [40, 24]. However, most of those methods choose either a half-half or leave-one-out cross validation scheme to split the training and testing sets. For example, in each round of the leave-one-out testing, 575 videos are used for training, and the remaining 24 videos are used for testing. Besides, for each video, there are $300 - 500$ frames in which the actor repeatedly performs one single action. If we assume one complete action lasts 30 frames, the actual training set for the above leave-one-out scheme contains at least 5750 samples, and for each action category, there are 960 samples. In many real-world applications, it is impossible to collect equivalently large training sets for any given action.

Weizmann dataset: The Weizmann human action dataset contains 93 sequences of nine actors performing ten different actions. Each sequence contains about $40 - 120$ frames. In the figure-centric representation, all frames have been normalized to 90×60. The best performance published is 100% by using the large training set [16].

Cluttered human action dataset: The cluttered human action dataset is a variant of the dataset collected by Ke et al. [26], which was initially designed for action detection in the

Figure 3.4: Sample frames of cluttered human action dataset [26].

crowed environment. It contains not only cluttered static backgrounds, but also cluttered dynamic backgrounds, such as moving cars or walking people. In order to test the robustness of our action recognition methods, we use it for recognition. From each raw video sequence in the original dataset, we manually crop out the actions of interest. This dataset contains 95 sequences with five actions, jumping jack, pushing elevator button, picking-up, one-hand waving, and two-hand waving. Each sequence contains about $30-50$ frames. Representative frames are shown in Fig. 3.4.

3.3.2 Experimental results

We perform the following experiments to evaluate our patch based comparison method and the transferable distance function learning:

1. Evaluate the patch based comparison method on all three datasets;
2. Train the transferable distance function on Weizmann, and test on KTH;
3. Train the transferable distance function on KTH, and test on the cluttered action dataset.

Direct Comparison on KTH. In this experiment, we evaluate the patch based direct comparison method on the KTH dataset. We first randomly select one actor, then randomly choose one clip per action from this actor as the template set. The clip contains at most 25

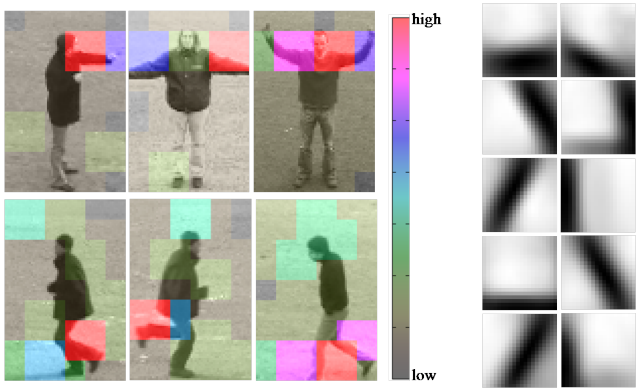
frames, i.e. $1 - 1.5$ complete action cycles. The sequences of the remaining actors are used as the query set. We decompose each frame into 40 patches. The patch size is 20×20 and the length of strides is 10.

We run the experiment five times and for each round we select a different actor as the template. The results are shown in the row of **Dc** of Table 3.1. The average result over the five rounds is 72.48%. Although the performance of our patch based direct comparison method is inferior to the previously published results, as shown in Table 3.2, our method requires much less training data. Note that due to the action variation in person, the performance depends on how distinguishable the templates are.

Training on Weizmann and Testing on KTH. In this experiment, we train a transferable distance function from Weizmann and test it on KTH. In order to meet the requirement of the transfer learning scenario, i.e. the source training set does not contain the actions of the template set, we remove walking, running, and two-hand waving from the Weizmann dataset. We build the codebook vocabulary on the remaining sequences of Weizmann as described in Section 3.2.4. The number of codewords is set to 100. We used other codebook sizes and found they do not affect the performance substantially. In training, the parameters are set as, $\sigma = 0.5$ and $C = 0.0001$. Through training on Weizmann, we can obtain the relation $\mathbf{P}$, which parameterizes the transferable distance function .

After the training, we first compute the hyper-features of the query videos in KTH using the codebook constructed from Weizmann. Then, we can obtain the distance function through Eqn. 3.3. For the purpose of illustration, we visualize the learned weights in Fig. 3.5(a). The red patches refer to high weights. Note that patches on the frames are overlapping, we only show the highest weight for an overlapping region. For the six actions in KTH, we can see most of the patches with high weights lie on the most important human parts, such as out-stretched arms or legs. Unlike other motion based interest point detection methods [33], the learned weight for the moving body is lower than moving legs. This is more intuitive since the moving body does not help to distinguish running, jogging and walking. Moreover, the learned $\mathbf{P}$ allows us to rank the visual words in the codebook vocabulary. We visualize the appearance feature of top ten words in Fig. 3.5(b). We can observe that these words are all "out-stretched-limb-like".

The recognition accuracies of five rounds of experiments are given in the row of **Tr** of Table 3.1. Note that for each round, we use the same templates as the direct comparison experiments. The largest improvement made by the transferable distance function is almost



(a) (b)

Figure 3.5: (a) Illustration of the learned weights on the six actions of KTH. (b) The learned **P** allows us to rank the visual words in the vocabulary. The top ten words are visualized. Note that our visual words consist of appearance and spatial location features. Only appearance is illustrated. Please refer to text for more details.

	1	2	3	4	5	Avg.	Std.
Dc	0.776	0.709	0.829	0.564	0.746	0.725	0.100
Tr	0.784	0.767	0.829	0.617	0.789	0.757	0.073

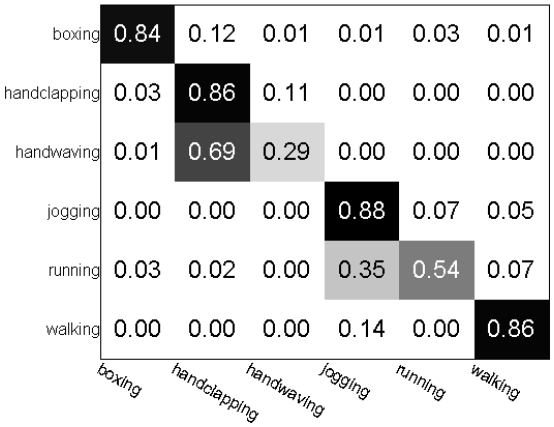
Table 3.1: The accuracy of five rounds of experiments on KTH. The top row denotes the round index. The row of **Dc** refers to the results of direct comparison, and the row of **Tr** refers to the results of training on Weizmann and testing on KTH. Std. denotes the standard deviation. Note that direct comparison is equivalent to the method using transferable distance function if setting all **w** to 1.

methods	accuracy	remark
Liu & Shah [31]	0.9416	LOO
Schindler & Van Gool [40]	0.9270	LOO
Jhuang et al. [24]	0.9170	Split
Nowozin et al. [35]	0.8704	Split
Neibles et al. [33]	0.8150	LOO
Dollar et al. [10]	0.8117	LOO
Ours (**Tr**)	**0.7571**	**One clip**
Ours (**Dc**)	**0.7248**	**One clip**
Schuldt et al. [41]	0.7172	Split
Ke et al. [25]	0.6296	Split

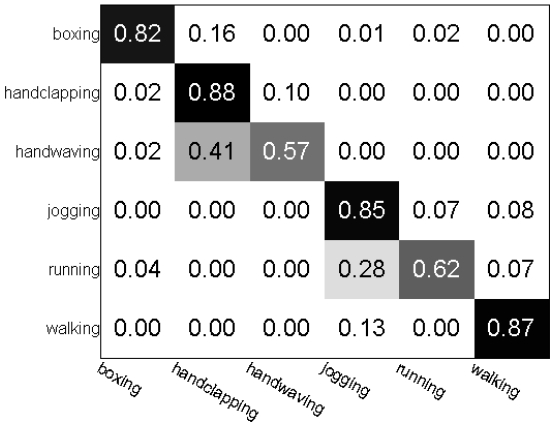
Table 3.2: Comparison of different reported results on KTH. We remark the setup of the training set. LOO refers to the "Leave-one-out" cross validation. Split refers to other split strategies of training and testing sets. Note that these numbers are not directly comparable due to variations in training/testing setup.

6%. We can observe that in experiment round 1 and 3, the improvements made by the transferable distance function are minor. This is reasonable since the direct comparison has already achieved very good results. We also show the confusion matrices of experiment round 2 in Fig. 3.6. We can see that the transferable distance function significantly mitigates the confusion of the most difficult actions, such as hand-clapping versus hand-waving, and jogging versus running. In particular, we see an improvement of almost 30% for the hand-waving. The comparison with previously published results are given in Table 3.2.

Another benefit of learning transferable distance function is that it can be used to speedup the comparison. In the patch based direct comparison method, for each patch on the query frame, we need to search its corresponding area on the template frame and find the best matched one. This process is time-consuming since there exist 1000 patches over the sequence of 25 frames. With learned distance function of the query sequence, we can sort the patches on each frame by their weights. Instead of using all patches for matching, we only choose the top N patches with high weights from each frame. We change N from 1 to 40 and compute the average accuracy over the five rounds of experiments. The results are illustrated in Fig. 3.7 (a). Using only ten patches on each frame, we can achieve a better result than the patch-based direct comparison using all patches on the frame. This would save 3/4 matching time, and significantly increase the efficiency of whole recognition process.

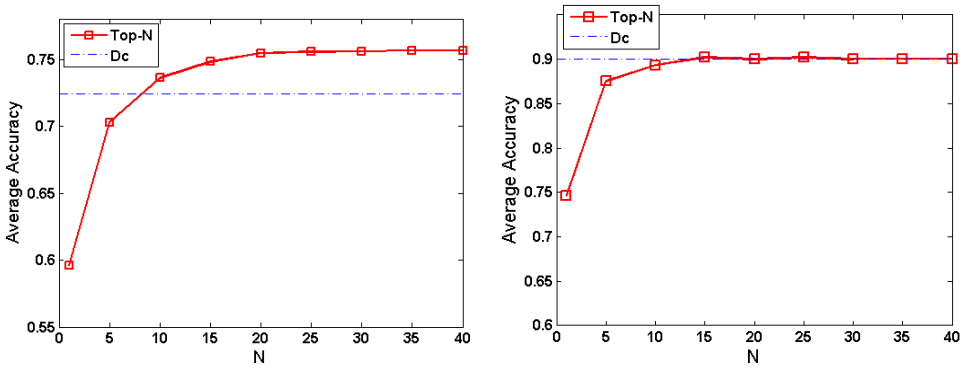


(a)



(a)

Figure 3.6: Confusion matrices on KTH of experiment round 2. Horizontal rows are ground truths, and vertical columns are predictions. (a) Direct comparison. (b) Training on Weizmann and testing on KTH.



(a) (b)

Figure 3.7: (a) The average accuracy of five rounds of experiments on KTH using only top N patches of each frame; (b) The average accuracy of five rounds of experiments on cluttered action dataset using only top N patches on the frame. The dash-dot line denotes the average accuracy of the direct comparison using all patches.

	1	2	3	4	5	Avg.	Std.
Dc	0.928	0.892	0.916	0.819	0.795	0.870	0.060

Table 3.3: The accuracy of five rounds of experiments on Weizmann using patch based direct comparison. The top row denotes the round index. Std. denotes the standard deviation.

Direct Comparison on Weizmann The setup we use in this experiment is exactly the same as the direct comparison experiment on KTH. In each round of the experiment, we randomly select one actor and use one clip per action with 25 frames from this actor as the template. The sequences of the remaining actors are used as the query set. The results are shown in Table 3.3. We compare our results with the work of Tran and Sorokin [43], as shown in Table 3.4. Our result outperforms both "1-Nearest Neighbor + motion context descriptor (1NN)" and "1-Nearest Neighbor with metric learning + motion context descriptor (1NN-M)". Note that we only use a 25 frame clip as the template rather than the whole video as in [43].

Unfortunately, a fair transfer learning experiment training on KTH and testing on Weizmann is not possible. After removing overlapping actions, there are only three actions left in the KTH (boxing, hand-clapping and jogging). The number of actions is too small to contain enough generic knowledge. So we do not run the experiments of training on KTH and testing on Weizmann.

	Dc	1NN [43]	1NN-M [43]
FE-1	0.8699	0.5300	0.7231

Table 3.4: Comparison of the average accuracy on Weizmann using one exemplar per action with [43].

	1	2	3	4	5	Avg.	Std.
Dc	0.944	0.900	0.844	0.900	0.911	0.900	0.036
Tr	0.944	0.900	0.856	0.900	0.900	0.900	0.031

Table 3.5: The accuracy of five rounds of experiments on the cluttered human action dataset. The top row denotes the round index. Std. denotes the standard deviation. Note that direct comparison is equivalent to the method using transferable distance function if setting all **w** to 1.

Direct Comparison on cluttered action dataset. The goal of this experiment is to evaluate the robustness of our patch based direct comparison on more challenging datasets with cluttered backgrounds. For each action, we randomly choose one clip with 25 frames as the template and the remaining sequences as the query set. The same patch decomposition scheme is used. Similarly, we perform five rounds of experiments by choosing different templates. The results are shown in the **Dc** row of Table 3.5. We can see the patch based direct comparison achieves very high accuracy on this dataset.

Training on KTH and testing on cluttered action dataset. This experiment follows the same protocol as training on Weizmann and testing on KTH. We first remove the two-hand waving action from KTH since it also exists in the cluttered action dataset. KTH contains a large number of sequences, we choose only five actors' sequences to form the source training set. The results are shown in the **Tr** row of the Table 3.5. As expected, the transferable distance function learning achieves almost identical results as the direct comparison, since direct comparison has achieved very promising results. However, the transferable distance function can be used to sort the patches and choose the patches with top N highest weights, and thus improve the efficiency of the recognition system. As illustrated in Fig. 3.7(b), we are able to use only top 5 patches on each frame and achieve 86.67% accuracy. The efficiency is boosted significantly (saving 7/8 matching time) with the cost of only 3% accuracy decrease.

3.4 Summary

In this chapter we have presented an action recognition algorithm based on a patch-based matching scheme. A set of motion patches on input query clips and template clips with known actions is matched. This matching scheme proves to be effective for action recognition in the difficult case of only a single training clip per action. Further, we have demonstrated that learning a transferable weighting on these patches could improve accuracy and computational efficiency. These weights, based on patch hyper-features, are generic, can be directly applied to novel video sequences without further learning, and hold promise for recognition in small training set scenarios such as video retrieval and surveillance.

Chapter 4

Efficient Human Action Detection

This chapter addresses the problem of human action detection. Given a template video clip containing an actor performing a particular action, we would like to localize similar actions (in both time and space) in our test videos. In particular, we are interested in the scenario where the target action is specified using a *single video clip*. This is a natural and realistic scenario in many real-world applications, e.g., surveillance, video retrieval, etc.

In Chapter 3, a patch based matching scheme is used for action recognition with a single clip as the template. The *transferable distance function* has been proposed to weight those patches by their importance. The transferable distance function is learned from previously training sets, and can be applied to videos of new actions without further learning. In this chapter, our main goal is to address human action detection, which does not require the pre-processing human detection and tracking step on test videos as [11]. The main contributions of the work presented in this chapter are two-fold, in addressing the efficiency issues. First, we propose a variant of the motion feature in Efros *et al.* [11] using a histogram representation. This feature representation can be computed efficiently using integral images. Second, we propose a cascade structure for action detection with only one template, which is based on the transferable distance learning framework, and significantly boosts the efficiency of our approach.

4.1 Sliding window approach

Given a template action, the objective of human action detection is to localize all similar actions in test videos. In this paper, we choose the standard sliding-window approach,

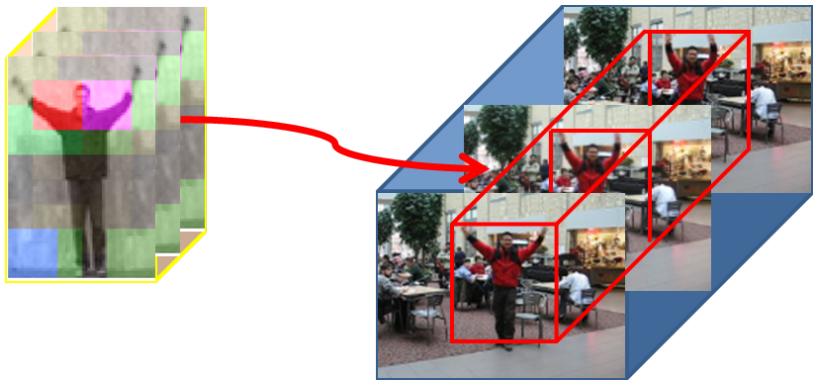


Figure 4.1: The illustration of the sliding window approach. The left video clip is the template T. The right video clip is the test video V, and the red bounding-box is the video segment L which is centered around location l.

that is to slide the template action video clip T over all locations on the test video V, as illustrated in figure 4.1. The distance between T and V at location l is denoted as $D(T, L)$, where L is the video segment of V centered around location l. An action is detected if the distance falls below a threshold. To compute the distance $D(T, L)$, we choose the patch-based action comparison approach as discussed in Section 4.1.2. In order to further enhance the efficiency of action detection, we choose a histogram representation of four-channel motion flow feature.

4.1.1 Histogram based motion feature

Our motion feature is a variant of the descriptor proposed by Efros *et al.* [11] which has been widely used in action recognition. First, we compute the optical flow at each frame, then split the optical flow vector field F into the horizontal and vertical components, F_x and F_y. They are further half-wave rectified into four non-negative channels F_x^+, F_x^-, F_y^+, F_y^-. Then, those four channels are blurred using a Gaussian kernel.

One of the limitations of this four-channel descriptor is its large size. For a small 20×20 patch, the dimensionality of the four-channel descriptor is $4 \times 20 \times 20 = 1600$. The distance between two feature vectors cannot be computed efficiently with such a high dimensional feature. In this paper, we break the patch into 4×4 cells. Each cell is represented by a four-bin histogram, where each bin corresponds to one channel in the four-channel motion descriptor [11]. The value of each bin is the accumulation of the weighted votes of all

pixels in the cell. In the end, we will obtain a feature vector with dimensionality only $4 \times 4 \times 4 = 64$. This motion feature is closely related to the histogram of optical flow used in [29]. The similarity between two feature vectors can be computed using Euclidean distance. Moreover, to efficiently compute feature vectors, the *integral image* representation [44] is used in the computing of each histogram bin.

4.1.2 Patch based action comparison

For the task of action detection, when using only one template, generalization is normally very difficult because of the intra-class variation among actors. In order to alleviate the effect of this variation, Ke *et al.* [26] manually break the template model into several parts over space and time. Instead, we use a simple patch-based approach that requires no manual interaction.

We compute distance $D(T, L)$ by comparing the patches from two video segments T and L. Each frame is decomposed into a number of 20×20 patches automatically, then $D(T, L)$ is computed as follows:

$$D(T, L) = \sum_{i=1}^{M} \sum_{s=1}^{S} \min_{r \in R_s} d(t_{is}, q_{ir}) \tag{4.1}$$

where t_{is} denotes the s-th patch on the template frame i, and q_{ir} denotes the r-th patch on the test frame i. R_s is the corresponding search region of s-th patch. M is the number of frames in a video segment. S is the total number of patches on each frame. $d(\cdot, \cdot)$ refers to the distance between two patches. For simplicity, we ignore the action speed variation between people, and directly correspond the frames from T to L in sequence. One could also apply dynamic programming based approaches to find the frame correspondence and thus alleviate the variation in speed.

4.2 Cascade structure

As in most object detection tasks, e.g. face detection and car detection, human action detection is a *rare event detection*. Hence, when using a window-scanning approach, it is important to efficiently reject the majority of negative sub-windows. Viola and Jones [44] proposed a cascade structure in the AdaBoost learning framework. Most of the negative sub-windows are rejected by simpler detectors efficiently, and then more complex detectors are applied to achieve low false positive rates. However, the training of boosted detectors

requires a large number of both positive and negative training samples. In the case of human action detection, it is difficult and even impossible to collect such a large training set for any given action. In particular, in our scenario, only one template is available for each action category.

In order to build a cascade structure with only one template, we use the *transferable distance function learning* proposed in Chapter 3. We first reiterate the terminology we will use. The *source training set* denotes the large dataset we already have at hand, for example a standard benchmark dataset (e.g. KTH). The *template* denotes the video we use to detect an action in test videos. Note that the source training set does not contain the same action as the template.

A key feature of the cascade structure is to use simpler but efficient detectors at the early stage to reject most negative sub-windows. The learned distance function provides us a useful tool to obtain such a simple detector. After learning on the source training set and obtaining the parameter $\mathbf{P}$, we are able to compute the weights (i.e. importance) of the patches on any given template action through Eqn. 4.2, based on their hyper-features. Then we can rank these patches by their importance. The details of transferable distance function learning can be found in Chapter 3.

$$w_i = \langle \mathbf{P} \cdot \mathbf{f}_i \rangle \tag{4.2}$$

At the early stage of the cascade structure, for the matching task, we can use only a subset of patches with high weights on the template video. For example, we can choose only two patches from each template frame with top-2 high wights at the first stage of the cascade structure. For a template video with 25 frames, only 50 patches are used at the first stage, so it could be very efficiently matched with all the sub-windows in test videos. The majority of negative sub-windows can be discarded after this stage. For the following stages, we can incrementally increase the number of patches utilized in the template video, and all patches will be used at the final stage in order to achieve an accurate matching. At the k-th stage of our cascade structure, distance $D^k(T, L)$ is computed as:

$$D^k(T, L) = \sum_{i=1}^{M} \sum_{s \in E_i^k} w_{is} \min_{r \in R_s} d(t_{is}, q_{ir}) \tag{4.3}$$

where E_i^k is the set of effective patches on the i-th frame at the k-th stage, and w_{is} is the weight assigned to the template patch t_{is}.

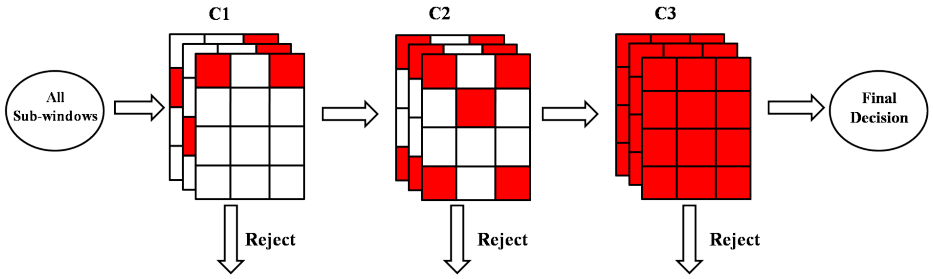


Figure 4.2: An example of the cascade structure. The red patches are the effective patches on template frames. At the **C1** stage, the top-2 patches of each frame with high weights are used to match with the input sub-windows. At the **C2** stage, top-5 patches are used for matching. At the final stage, all patches are used.

In the cascade structure of [44], the detection and false positive rates of each stage can be controlled using training and validation sets. However, in our scenario, only one template video is available for each action category, and there is no training dataset containing the same action as the template. Here we choose a rather simple way to control the performance of each stage. The detection threshold of a stage is set so that a certain number of sub-windows with high matching distances will be discarded. The remaining sub-windows will be evaluated by the next stage of the cascade structure. An example of the cascade structure is given in Fig. 4.2. Note that it is possible that early stages of the cascade structure may have high false negative rates and thus decrease the performance of whole structure. However, the experimental results in Section 4.3.2 demonstrate our cascade structure achieves similar results to the direct scanning method without using a cascade, which implies the early stages of our cascade structure can reliably keep the true positive sub-windows.

4.3 Experiments

We evaluate our method on the cluttered human action dataset collected by Ke *et al.* [26], and a ballet video sequence. We first review the human action datasets, then present the experimental results.

4.3.1 Datasets

Weizmann Dataset [6]: The Weizmann human action dataset is a standard benchmark human action dataset. It contains 93 sequences of nine actors performing ten different

actions. There are about $40 - 120$ frames for each sequences. This dataset is used as the source training set, so we choose the same *figure-centric* representation as [47]. After computing the motion feature, we crop each frame to 90×60 and put the human figure in the center of the frame.

Cluttered Human Action Dataset [26]: The cluttered human action dataset contains not only cluttered static backgrounds, but also cluttered dynamic backgrounds, such as moving cars and walking people. There are 48 videos containing 110 actions of interest. Each video contains approximately $300 - 800$ frames with resolution 120×160. Five types of actions are labeled: one-hand waving, two-hand waving, picking-up, pushing an elevator button, and jumping-jacks.

4.3.2 Experiments on the cluttered action dataset

For human action detection on the cluttered dataset, we first choose one template video for each labeled action event. Except for the action of pushing an elevator button, we use the sequences of the actor *ido* from the Weizmann dataset as templates. For the action of pushing an elevator button, we choose the template provided by Ke *et al.* [26]. Note that this selection of template videos increases the difficulty of the task since the template and test videos are captured under different instructions. All template videos contains only $20 - 25$ frames, i.e. $1 - 1.5$ complete action cycles.

The figure-centric representation is applied to template videos and all template frames are normalized to 90×60. Representative frames of template videos are shown in Fig. 4.3. After computing motion features, each frame is decomposed into 40 patches. The size of a patch is 20×20 and the length of the stride is 10.

To meet the requirement of the transfer learning scenario, in our experiments, the source training set does not contain the action of the template video. For example, in the experimental setup of jumping-jacks action, we remove the action of jumping-jacks from the Weizmann dataset. Then the remaining sequences form the source training set. After the training, we first compute hyper-features of the template video. Then, we can obtain the distance function of the template video through Eqn. 4.2. The detection of other actions follows the same experimental setup. Note that for the experiment of each action, the source training set does not contain the same action as template. The weights of the distance function are visualized in Fig. 4.3. As we can see, the high weights (red patches) are assigned to the *important* parts, such as the stretched-arm, and bent-back.

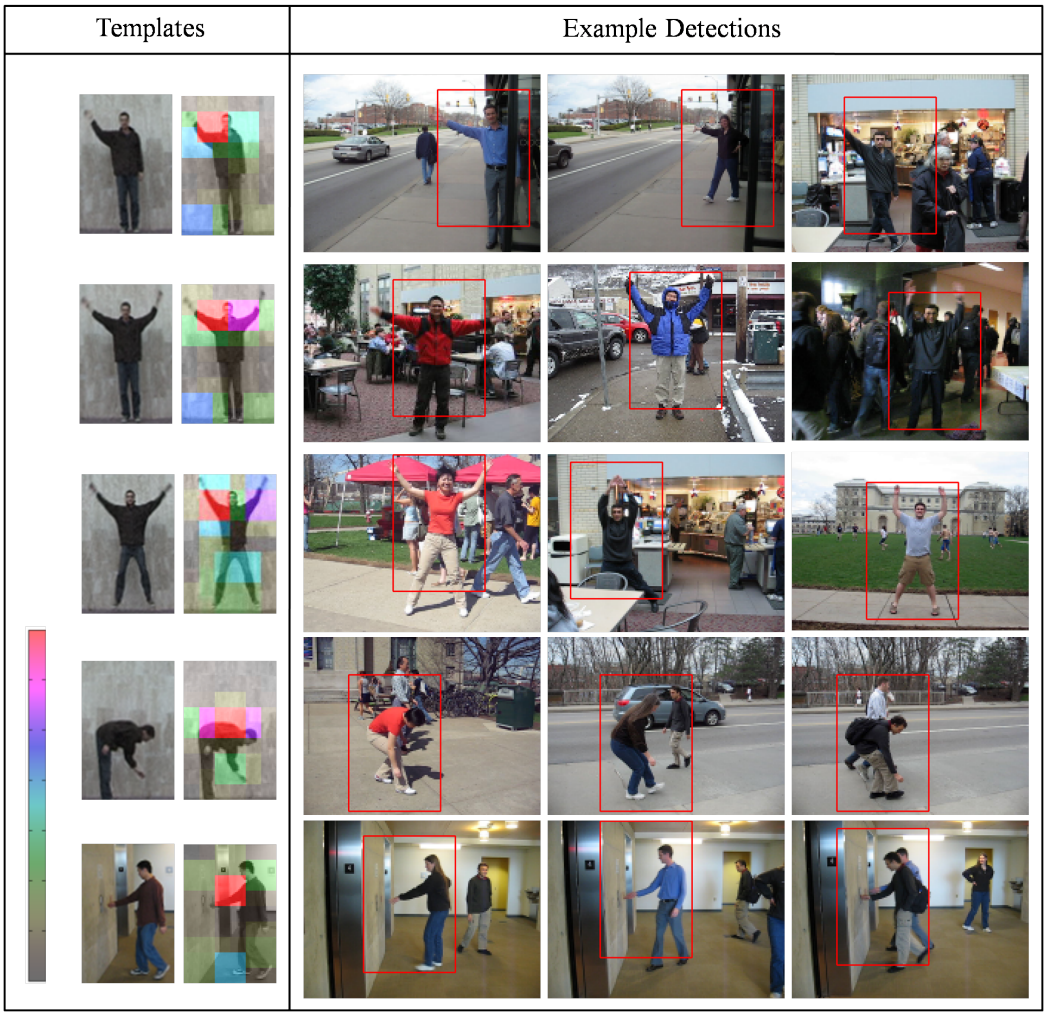


Figure 4.3: Action detection examples on the cluttered action dataset. Representative frames of the template videos and the visualization of learned weights are shown on the left. The left bottom corner shows the color bar for the visualization. Correct detection examples are shown on the right.

After training, we can build the cascade structure based on the learned distance function. In the experiments, the cascade structure consists of four stages. At the first stage, there are only two effective patches on each template frame. At this stage, the template video is scanned across the test video. Subsequent locations are obtained by shifting the template video either 5 pixels along the x or y axis, or 5 frames along the time axis. Similar to [26], the template videos are matched with the test video under a fixed scale. The speed of this stage is 20 times faster than using all patches on the template video. After the first stage, 90% of the sub-windows are set to be rejected. The second stage has five effective patches on each frame, and 80% of the remaining sub-windows from last stage will be rejected. For the third stage, ten patches on each frame are effective and 80% of the sub-windows will be kept at this stage. All patches on the template video are effective at the final stage. These parameters of the cascade structure are all the same for the experiments of each action.

Let N be the number of sub-windows on the test video. Let M be the number of frames of the template video, and each frame has been decomposed into 40 patches. Assuming the computing of patch-to-frame distance in Eqn. 4.3 takes time T, the computing time of directly scanning the template video over the test video without using cascade is $T_{nc} = 40MNT$. Under the above cascade setup, the computing time will be reduced to $T_c = 2M \times NT + 5M \times 0.1 \times NT + 10M \times 0.02NT + 40M \times 0.016NT = 3.34MNT$. There is an order of magnitude reduction in computing time, though the computational complexities of both methods are linear in the size of test videos.

The training of transferable distance function takes about several hours. However, the training session is done off-line. After obtaining the $\mathbf{P}$ from training, we can compute the weights of the patches on any given template action through Eqn. 4.2. Therefore, we only compare the running time in the test session. The methods are implemented in Matlab/MEX and run on a 2.40GHz Intel processor. There are 25 frames on the template video, and 800 frames on the test video. In order to build the cascade structure, we first compute the hyper-features of the template video, which takes 0.9 second. Then, we scan the template video over the test video using cascade struture, which takes only 30.3 seconds. However, it will takes 348.2 seconds without using the cascade.

Similar to [26], we project the obtained three-dimensional distance map to a one-dimensional vector of score. Only the best detection is kept for each test frame. The Precision-Recall curves are generated by changing the detection threshold, as shown in

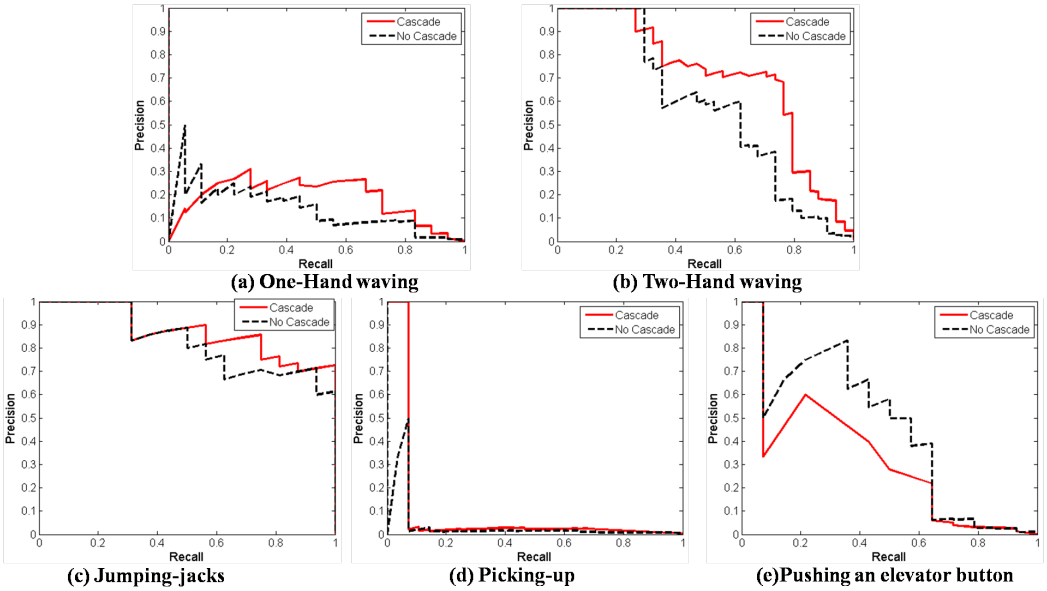


Figure 4.4: Precision-Recall curves of the action detection on the cluttered action dataset.

Fig. 4.4. Since we choose a different way to scan the template over test videos, our results are not directly comparable with [26]. We admit this dataset is very difficult because of the cluttered background. However, by only using the motion cue, our method is still able to achieve very good performance for jumping-jacks, two-hand waving, and pushing an elevator button. For the picking-up action, there is a large intra-class variation of actors performing the this action. The actor in the template clips performs this action by bending his back, but actors in the test videos usually perform the action by bending their keens. Therefore, our method achieves very low detection rates on this action. The sharp drop in the Precision-Recall curve is because there are a small number of test clips that match the style of the template (bending the back), but the rest are all different to the style of the template. One-hand waving is often confused with the two-hand waving and jumping-jacks and thus has a higher false positive rate. Example detections are shown in Fig. 4.3. From Fig. 4.4, we can also observe that, except for the action of pushing an elevator button, our cascade structure achieves better accuracy.

We give an example with more details in Fig. 4.5 about the detection of jumping-jacks in a video which contains some confusing actions, such as one-hand waving and two-hand waving. It is interesting to note that in the projected matching distance, the confusing

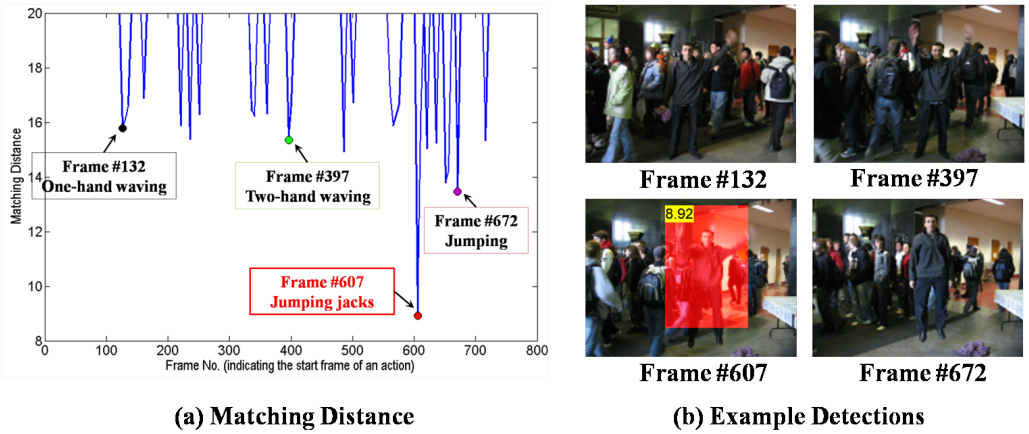


Figure 4.5: (a) Projected matching distance of the detection of jumping-jacks. (b) Example detections. The true positives are highlighted in Frame #607, where the left corner is the matching distance. The rest frames are all true negatives.

actions cause very low matching distances but they are still much higher than the jumping-jacks action.

4.3.3 Experiment on the ballet video

We apply our method to detect "spin" actions in a ballet video. Although this ballet video is very "clean", it contains more complex actions and two actors are performing the same actions in each frame. In addition, the actress wears a skirt and the appearance is very different to the template, which might cause difficulty for shape-based methods (e.g. [26]).

The Weizmann dataset serves as the source training set. The learned weights on the template video are visualized in Fig. 4.6(a). Note that the actions in the Weizmann dataset are distinctly different from the "spin" action of ballet. Our transferable distance function is still able to assign high weights onto the important parts such as the stretched-arms and legs. After training, we scan the template over the test video using the cascade structure. The matching distances of correct detections for the actor and actress are 2.31 and 5.81 respectively. Although the matching distance for the actress is higher than the actor because of the clothing, these distances are still much lower than any other portion of the video.

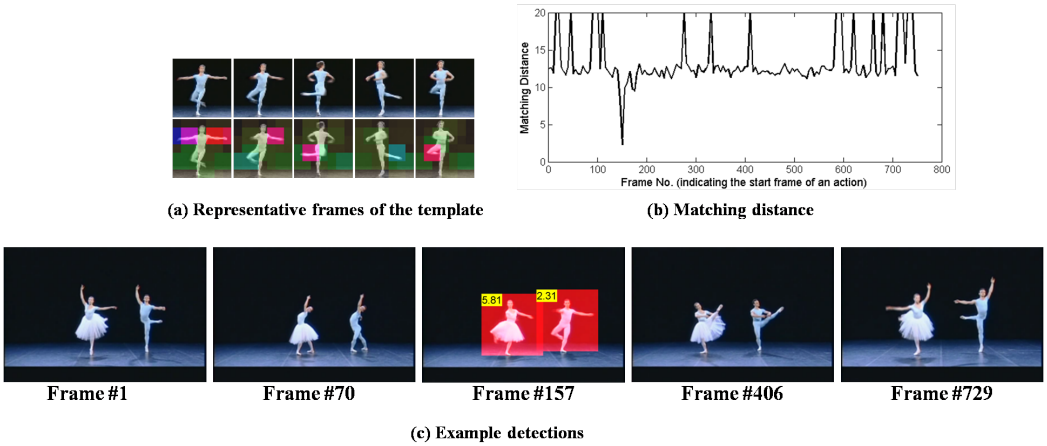


Figure 4.6: (a) Representative frames of the template videos, and the visualization of learned weights. (b) Projected matching distance. (c) Example detections. The true positives are highlighted in Frame #157, and the rest frames are all true negatives.

4.4 Summary

In this chapter, we have presented an efficient human action detection approach using only one template video. We have developed a histogram representation of the four-channel motion descriptors [11], which can be efficiently computed using integral images. Based on the learning of a transferable distance function, a cascade structure has been proposed. Experimental results show that our cascade structure achieves reliable detection results and improves the efficiency of the patch based action detection method significantly.

Chapter 5

Conclusion

In this thesis we have presented a novel transfer learning technique called transferable distance function learning. We have applied it in human action recognition and demonstrated the efficiency and effectiveness of this method. For the task of human action detection, we have proposed a cascade structure based on the learned transferable distance function which can significantly improve the efficiency of the patch-based action detection system. In the following, I will briefly highlight the limitations of the transferable distance function learning and the future research.

5.1 Limitations

In Chapter 3, we have demonstrated transferable distance function learning can improve both accuracy and efficiency of the patch-based template matching algorithm on several different datasets. However, it is also very important to recognize the limitations of transferable distance function learning. Similar to other transfer learning techniques, one of major limitations of transferable distance function learning is its limited power of knowledge transfer. The success of transferable distance function requires that there is some common knowledge shared between *source training set* and *template set* and this knowledge can help improve the performance on the template set. Therefore, an important issue in transferable distance function learning is "when to transfer", which is to address under which situation the above requirement will be met and transfer learning can be done. In some situations, for example the knowledge transferring from "recognizing a face" task (*source*) to "recognizing an action" task (*target*), there is almost no common knowledge shared between them. The

brute-force knowledge transfer between these two tasks is nonintuitive and unnecessary. In the worst case, the performance on the target task may decrease due to the knowledge transferred from source task, which is referred to a situation — *negative transfer*[37].

The transferable distance function learning can be applied to various computer vision problems *i.e.* face recognition, object recognition and identification. For those particular computer vision applications, we would like to clarity in which situations transferable distance function learning can help improve the performance of target task. First of all, the source and target domains must be similar. For example, we may transfer knowledge from one action category to the other one. But there is little knowledge can be transferred from a face to an action category. Second, the learning of transferable distance functions is to extract *generic* knowledge of patch weighting from source training set. Intuitively, the source training set needs to consist of a large enough number of action categories, in order to represent a generic action knowledge. In our experiments, we have shown by only using seven categories in the source training, the performance in template set can be improved by transferable distance function. We do believe by increasing the number of category in the source training set, the performance can be future improved. But it is unclear how many categories for source training set would be enough, and the analysis of knowledge transferability between tasks is also an open issue in machine learning community[37]. We would like to explore this issue in the future when large annotated action datasets are available.

5.2 Future work

In this thesis, we assume there exists a certain relationship between the importance and the hyper-feature of a patch. Then, the weight to be assigned to each patch can be computed by $w_i = \langle \mathbf{P} \cdot \mathbf{f}_i \rangle$. The parameter $\mathbf{P}$ is learned by a max-margin learning framework. Although the hyper-feature $\mathbf{f}_i$ includes the information of patch locations, this formulation still implies a strong independence between each patch. However, one example of the advantage of patch dependence is that for the same "stretched-arm-like" patches, the patches appearing near to the "head-like" patch would be much more important than other "stretched-arm-like" patches. Because with cluttered background, some strong edges in the background may look like "stretched-arm". Therefore, in our future research, we would like to take into account the dependency information between patches when we compute the hyper-features for a video clip.

In this thesis, we claim that we can construct the "source" training set from the standard benchmark datasets, such as KTH [41] and Weizmann [6] benchmark datasets. The number of video clips in those datasets is still very small, compared with the enormous number of videos on the Internet (YouTube). However, those YouTube videos are not well labeled. Therefore, in order to exploit more information from the Internet to assist the task of action recognition from a single clip, in our future research, we would like to incorporate the unlabeled data into our learning framework. That promises to be a very novel combination of semi-supervised learning and supervised learning.

In conclusion, we have proposed transferable distance function learning in this thesis, and applied it in human action recognition and detection. We truly believe transfer learning holds promise for improving many computer vision applications.

Bibliography

[1] Amr Ahmed, Kai Yu, Wei Xu, Yihong Gong, and Eric P. Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *European Conference on Computer Vision*, 2008.

[2] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Mult-task feature learning. In *Advances in Neural Information Processing Systems*. MIT Press, 2006.

[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[4] Andreas Argyriou, Charles A. Micchelli, Massimiliano Pontil, and Yiming Ying. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems*. MIT Press, 2007.

[5] Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *Proceedings of Computational Learning Theory*, 2003.

[6] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *IEEE International Conference on Computer Vision*, 2005.

[7] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[8] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *International Conference on Machine Learning*, 2007.

[9] Navneet Dalal and Bill Triggs. Histogram of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[10] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *ICCV'05 Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[11] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision*, pages 726–733, 2003.

[12] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *ACM SIGKDD*, 2004.

[13] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[14] Ali Farhadi, David Forsyth, and Ryan White. Transfer learning in sign language. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[15] Ali Farhadi and Mostafa Kamali Tabrizi. Learning to recognize activities from the wrong view point. In *European Conference on Computer Vision*, 2008.

[16] Alireza Fathi and Greg Mori. Action recognition by learning mid-level motion features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recongition*, 2008.

[17] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, April 2006.

[18] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[19] Andras Ferencz, Erik Learned-Miller, and Jitendra Malik. Learning to locate informative features for visual identification. *International Journal of Computer Vision*, 77(1-3):3–24, 2008.

[20] Andrea Frome. *Learning Distance Functions for Exemplar-Based Object Recognition*. PhD thesis, University of California at Berkeley, 2007.

[21] Andrea Frome, Yoram Singer, and Jitendra Malik. Image retrieval and classification using local distance functions. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

[22] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *IEEE International Conference on Computer Vision*, 2007.

[23] Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.

[24] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *IEEE International Conference on Computer Vision*, 2007.

[25] Yan Ke, Rahul Sukthankar, and Martial Hebert. Efficient visual event detection using volumetric features. In *IEEE International Conference on Computer Vision*, volume 1, pages 166–173, 2005.

[26] Yan Ke, Rahul Sukthankar, and Martial Hebert. Event detection in crowded videos. In *IEEE International Conference on Computer Vision*, 2007.

[27] Yan Ke, Rahul Sukthankar, and Martial Hebert. Spatio-temporal shape and flow correlation for action recognition. In *Visual Surveillance Workshop*, 2007.

[28] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[29] Ivan Laptev and Patrick Pérez. Retrieving actions in movies. In *IEEE International Conference on Computer Vision*, 2007.

[30] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI Conference on Artificial Intelligence*, 2008.

[31] Jingen Liu and Mubarak Shah. Learning human actions via information maximization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recongition*, 2008.

[32] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaption with multiple sources. In *Advances in Neural Information Processing Systems*. MIT Press, 2008.

[33] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In *British Machine Vision Conference*, volume 3, pages 1249–1258, 2006.

[34] Eric Nowak and Frederic Jurie. Learning visual similarity measures for compariing never seen objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[35] Sebastian Nowozin, Gokhan Bakir, and Koji Tsuda. Discriminative subsequence mining for action classification. In *IEEE International Conference on Computer Vision*, 2007.

[36] Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems*. MIT Press, 2009.

[37] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. Technical Report HKUST-CS08-08, Department of Computer Science and Engieerning, Hong Kong University of Science and Technology, 2008.

[38] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Transfer learning for image classification with sparse prototype representations. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[39] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *International Conference on Machine Learning*, 2007.

[40] Konrad Schindler and Luc Van Gool. Action snippets: How many frames does action recognition require? In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[41] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local SVM approach. In *IEEE International Conference on Pattern Recognition*, volume 3, pages 32–36, 2004.

[42] Eli Shechtman and Michal Irani. Space-time behavior based correlation. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

[43] Du Tran and Alexander Sorokin. Human activity recognition with metric learning. In *European Conference on Computer Vision*, 2008.

[44] Paul Viola and Michael Jones. Robust real-time object detection. In *Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling(SCTV)*, 2001.

[45] Daniel Weinland and Edmond Boyer. Action recognition using exemplar-based embedding. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[46] Weilong Yang, Yang Wang, and Greg Mori. Efficient human action detection using a transferable distance function. In *Asian Conference on Computer Vision*, 2009.

[47] Weilong Yang, Yang Wang, and Greg Mori. Human action recognition from a single clip per action. In *ICCV Workshop on Machine Learning for Vision-based Motion Analysis*, 2009.

[48] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *International Conference on Machine Learning*, 2005.

[49] Shipeng Yu, Volker Tresp, and Kai Yu. Robust multi-task learning with t-processes. In *International Conference on Machine Learning*, 2007.

[50] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report TR1530, University of Wisconsin at Madison, 2005.