Region-sequence based six-stream CNN features for general and fine-grained human action recognition in videos

Ma, Miao; Marturi, Naresh; Li, Yibin; Leonardis, Ales; Stolkin, Rustam

DOI:
10.1016/j.patcog.2017.11.026

Document Version
Peer reviewed version

Citation for published version (Harvard):
Ma, M, Marturi, N, Li, Y, Leonardis, A & Stolkin, R 2018, 'Region-sequence based six-stream CNN features for general and fine-grained human action recognition in videos', *Pattern Recognition*, vol. 76, pp. 506-521. https://doi.org/10.1016/j.patcog.2017.11.026

Region-sequence based six-stream CNN features for general and fine-grained human action recognition in videos

Miao Ma, Naresh Marturi, Yibin Li, Ales Leonardis, Rustam Stolkin

PII: S0031-3203(17)30478-8
DOI: 10.1016/j.patcog.2017.11.026
Reference: PR 6379

To appear in: *Pattern Recognition*

Received date: 4 May 2017
Revised date: 31 October 2017
Accepted date: 19 November 2017

Please cite this article as: Miao Ma, Naresh Marturi, Yibin Li, Ales Leonardis, Rustam Stolkin, Region-sequence based six-stream CNN features for general and fine-grained human action recognition in videos, *Pattern Recognition* (2017), doi: 10.1016/j.patcog.2017.11.026

Region-sequence based six-stream CNN features for general and fine-grained human action recognition in videos

Miao Ma[1,2,3,‡], Naresh Marturi[2,4], Yibin Li[3], Ales Leonardis[2], and Rustam Stolkin[2]

[1] *Qingdao University, Qingdao, Shandong, 266071, PR China;*
[2] *University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK;*
[3] *Shandong University, Jinan, Shandong, 250061, PR China;*
[4] *KUKA robotics UK Ltd., Wednesbury Great Western Street, WS10 7LL, UK;*
[‡] *Corresponding author: mxm443@cs.bham.ac.uk*

Abstract

This paper addresses the problems of both general and also fine-grained human action recognition in video sequences. Compared with general human actions, fine-grained action information is more difficult to detect and occupies relatively small-scale image regions. Our work seeks to improve fine-grained action discrimination, while also retaining the ability to perform general action recognition. Our method first estimates human pose and human parts positions in video sequences by extending our recent work on human pose tracking, and crops different scaled patches to obtain richer action information in a variety of different scales of appearance and motion cues. We then utilize a Convolutional Neural Network (CNN) to process each such image patch. Instead of using the output one dimension feature from the full-connection layer, we utilize the outputs of the pooling layer of CNN structure, which contains more spatial information. Then the high dimension of the pooling features is reduced by encoding, to generate the final human action descriptors for classification. Our method reduces feature dimension while also effectively combining appearance and motion information in a unified framework. We have carried out empirical experiments using two publicly available human action datasets, comparing the human action recognition result of our algorithm against six recent state-of-the-art methods from the literature. The results suggest comparatively strong performance of our method.

Keywords: human pose, action recognition, video understanding

1. Introduction

Video sequences provide much richer information about actions, compared to an individual still image. Consider a single still image showing a man holding a knife in a kitchen scene. We cannot tell what he is doing with the knife. Does he want to cut something? Or is he cleaning the knife? In contrast, such action understanding is often much more easily obtainable from a video sequence. Still images provide spatial cues [1] and provide information to answer "what is that?"; while video provides both spatial and temporal cues [2], and can answer "what is going on?".

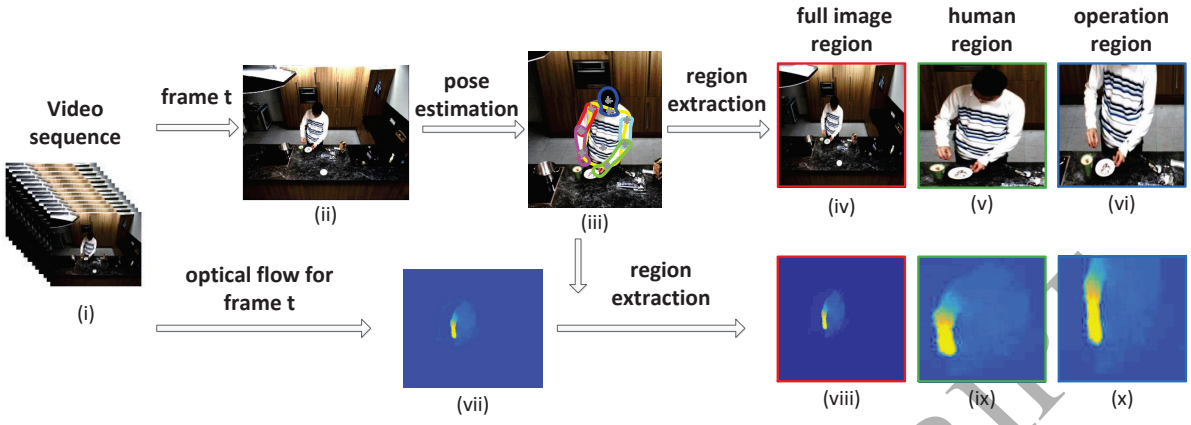
Recognition of human-induced actions in videos has gained significant amount of interest in the fields of computer vision and pattern recognition. This is due to its increasingly large number of applications in the areas of human-machine interaction [3], intelligent space[4], virtual reality, elderly care [5], robotics [6] etc. It also plays a vital role in many computer vision tasks such as video annotation, video retrieval etc. Human action recognition, which is a task of assigning videos to a set of action classes, is a challenging problem due to: large variety of activities [7], complex human actions and background movements [8], various observation views [2] and limited observation capacities [9], as well as ambiguous movements of different actions [10]. It has been intensively studied in the literature for more than two decades, where the corresponding methods varies from template matching [11], hand-crafted points of interest features [8], to deep learning methods [12].

While a variety of methods have been proposed in the recent years to recognize actions, most of them focus on videos with coarse actions [13, 14], such as lifting: where upper body moves upward; diving: where entire body drops; kicking: where one leg moves while the other remains static; etc. However, these are not typical in general scenes of life. In many applications, fine-grained actions [7] need to be recognized, e.g. washing hands versus falling water from tap. In this paper, we primarily focus on this problem of recognising fine-grained actions in the videos. It is highlighted that, for fine-grained action recognition, spatial regions that contain contextual cues have most distinguished information and should receive greater attention. Some related works in the literature [15, 16] use thousands of region proposals to extract action information, and then choose the most distinguished region features for action recognition. However, these methods can suffer in cluttered scenes, e.g. an image of a person squeezing an orange in the kitchen, with a stirrer nearby, might be erroneously recognised as stirring rather than squeezing. We estimate coarse pose along videos, and then extract the human body region and operation region in each frame, in order to enhance the effective pixels for human action. Then the enhanced patches are processed with CNNs.

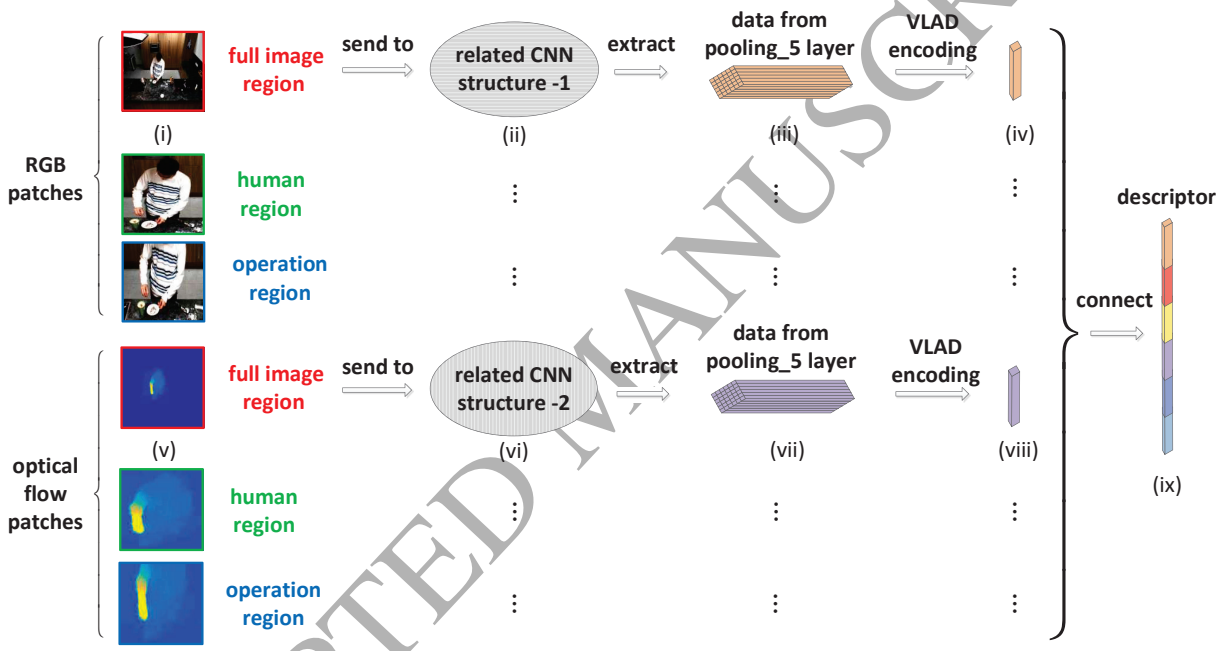
CNN structure contains convolutional layers, pooling layers, and fully connected layers. In convolutional and corresponding pooling layers, the kernel traverse all over the image with specified size and stride, which makes the output data maintain some spatial information. However, fully connected layer transforms previous multi-dimension data into one dimension vector, which would change and loose the spatial information mentioned in previous layers. That is to say, the last pooling layer data of CNN structure contains more spatial information than generally used full connection layer data. In our work, we propose an effective encoding method for the pooling layer data, which is able to make better use of spatial information and help to obtain more distinguishable descriptors for general and fine-grained human actions.

1.1. Overview of our method

In this section we provide an overview of the proposed method, highlighting important steps. Given a video, the following steps are performed to recognise fine-grained actions: firstly, we introduce a pose estimation method, which evaluates pose candidates based on appearance information and motion information,



(a) From video sequence to image patch sequences.



(b) From image patch sequences to fine-grained action descriptor.

Figure 1: Illustration of the proposed method pipeline. (a) Raw video sequence as system input, and then human pose detector is used for each frame to help segment all image region, human region and operation region patches for both color images and optical flow images. (b) Each image patch is processed by related CNN structure, and data from $pooling_5$ layer are encoded to obtain final fine-grained action descriptor.

and then we use the human pose estimation method to obtain the human body positions and regions in each video frame. Secondly, six image patch sequences are obtained based on the human body positions, and these patches are later fed to CNN structures as inputs. Finally, we construct human action discriminative features for videos by encoding pooling layer outputs of the CNN structures.

A detailed pipeline of our method is shown in Fig. 1. The main contributions of this work are explained as follows:

(i) Firstly, we propose a coarse human pose tracking and estimation method by extending our recent work on human pose tracking [17], see steps (i), (ii) and (iii) of Fig. 1(a). Unlike in our previous work, for recognizing human actions, we are not interested in the very exact locations of human keypoints; but aim at obtaining human body foreground regions and the corresponding appearance and motion cues to distinguish different human actions. In this case, our proposed human coarse pose tracker focuses on achieving the continuity and consistency of human foreground region in the video.

(ii) Secondly, we propose a method to use six image patch sequences to enhance and extract different scales and types of information for both appearance and motion cues. RGB and optical flow image sequences are used for obtaining appearance and motion information separately. The tracked human poses (from step (iii) of Fig. 1(a)) are used to crop regions from both RGB and optical flow images (see steps (ii) and (vii) of Fig. 1(a)). For each type of image, we get patches that contain human foreground regions (see steps (v) and (ix) of Fig. 1(a)), and regions around human arms (see steps (vi) and (x) of Fig. 1(a)). As a result, six image patch sequences are obtained for each video sequence. Each image patch has been resized to be of the same size *i.e.*, 224×224 pixels. The obtained patches enhance and insert effective pixels for recognizing fine-grained human actions.

(iii) Thirdly, instead of using the fully connected layer outputs of CNN structures, we propose a feature constructing method by encoding the outputs of last pooling layers (see steps (iii) and (vii) of Fig. 1(b)) using the vector of locally aggregated descriptors (VLAD) encoding method (see steps (iv) and (viii) of Fig. 1(b)). The last pooling layer data contain more spatial information than that of the fully connected layer, and some of the spatial information would be lost in the following full connection operation. Consequently, in our proposed method, we utilize data and information from the last pooling layer for patches of different types and scales. The proposed encoding and assembling method makes better use of multiple types of action information.

We test our method on two publicly available datasets: sub-JHMDB dataset and MPII Cooking dataset, and the results are compared against six other state-of-the-art algorithms.

The remainder of the paper is organized as follows. The methods that are closely related to our work are presented in Section 2. The proposed method for obtaining contextual cue regions is presented in Section 3. Experiments performed to validate the proposed method are discussed in Section 4. Section 5 provides concluding remarks and suggestions for future work.

2. Related Work

Human action recognition is a key research area in the field of computer vision, and has been previously surveyed by many researchers [18, 19]. In this section, we discuss some of the most relevant related work.

In the past decade, local features such as SIFT [20], HOG [21], HOF [22] etc. have been widely used for accomplishing visual recognition tasks [23, 24]. These methods often firstly extract spatial and temporal local features, and then use encoding methods, for example, bag of features (BoF) [25] to encode local features into vectored collections, and finally a classifier such as SVM is used for classification. However, each kind of feature is only able to describe a single property, such as color, contour or salient points. As a result, researchers rely more and more on combining several features together [26, 27] to represent complex properties. For instance, from their survey on multi-view learning, Xu *et al.* [28] identified that multi-view learning is rendered more effectively by exploring the consistency and complementary properties of different views. Later in [29], they handled the incomplete-view problem for image restoration by exploiting the connections between multiple views. Similarly, Li *et al.* [30] solved the image re-ranking problem by exploiting the complementarity between the deep features and shallow representations, and by integrating these two heterogeneous features into a multi-view feature learning model. Furthermore, Bregonzio *et al.* [31] fuse local pace and time individual descriptors with global spatio-temporal distribution information to solve action recognition problem. It is clear that these combined features improve the visual recognition accuracy compared with single feature methods.

In other cases, hierarchical structures [32, 33, 34] have been studied for creating more sophisticated hand-crafted features. For instance, Ma *et al.* [33] proposed a hierarchical structure for action recognition, where video segmentation trees are computed in the first layer using Ultrametric Contour Map (UCM) [35]. These trees are then pruned in the second layer using shape, color, motion, and other information. Later in the third layer, the method tracks the remaining segment trees both forwards and backwards in time, and used the bag-of-words representation for recognition. Liu *et al.* [36] also proposed a hierarchical structure for action recognition. In the first layer of their method, they combine optical flow with a biologically inspired feature to create a distinguishable feature, which is denoted as Pyramidal Motion Feature (PMF); then in the second layer, the PMFs are combined with spatial information to obtain final action descriptor.

Convolutional Neural Networks (CNNs) firstly emerged with the proposition of visual nervous system by Hubel and Wiesel [37], and were later implemented by Fukushima [38]. Since then, many researchers made efforts to improve the performance of CNN structures for various tasks. Wu *et al.* [39] proposed a kind of quantized CNN (Q-CNN) framework, which simultaneously speeds up the computation and reduces the storage and memory overhead of CNN models. It uses an effective training scheme to suppress the accumulative error while quantizing the whole convolutional network. Ijjina *et al.* [40] proposed a genetic algorithm based CNN (GA-CNN) structure, which uses solutions generated by genetic algorithms to initialize the weights of CNN classifier. The proposed GA-CNN structure is able to combine the global optimization capabilities of genetic algorithms with the local optimization ability of gradient descent algorithm, and as a result, it minimizes the classification error and improve the performance. Wang *et al.* [41] proposed a CNN

packing framework (CNNpack), which handles convolutional filters in the frequency domain using discrete cosine transform to compress neural networks. The CNNpack has high compression ratio and speed-up ratio proofed by experiments, which creates a bridge to link traditional signal and image compression with CNN compression theory.

Recently, CNNs have been used in a wide range of applications in different fields including computer vision, e.g. object detection [15, 42] and image classification [43]. Girshick *et al.* [15] constructed region-based CNN (R-CNN) features for detecting objects in still images, and the regions are pre-detected multiple parallel regions which are possible to contain distinguishable object information. Later, Gkioxar *et al.* [16] extended multiple parallel regions in [15] to become one primary region which is detected by a human detector and a set of secondary regions for human action recognition problem in static images. Lin *et al.* [44] proposed a bilinear CNN model for fine-grained object recognition in still images. In their work, they have used two similar CNN architectures for one RGB frame and calculated the outer product of the two CNN output matrices to obtain a vector descriptor for object recognition in static images. Yang *et al.* [3] made use of a two-stream CNN structure for recognizing human grasp actions in videos. In that work, one CNN stream was used for classifying the hand grasp type and the other for object recognition. Karpathy *et al.* [12] used a two stream CNN structure for video classification: one stream for low-resolution images and another for high-resolution center regions. Simonyan and Zisserman [2] extended the two stream CNN structure for action recognition in videos, but used one stream for color images and the other for optical flow images. This structure improved the action recognition accuracy by making use of both spatial and temporal information. Cheron *et al.* [10] extracted image patches around estimated human pose joints for CNN processing, showing that knowledge of human pose contains useful information about human actions. They also revealed that the accuracy of detected human poses plays an important role. However, human pose estimation and tracking is another challenging problem in its own right [17].

Inspired by region-based methods for still images [15, 16], we propose a region-based action recognition method for videos. We propose a coarse human pose detector, which estimates human poses in each frame of the video to identify human foreground regions. Additionally, we define an operation region for humans who are performing fine-grained actions. Heretofore, we achieve six-stream patch sequences that are then processed by the CNN structures. We use two types of CNN structures: one for the three RGB streams, and the other for the three optical flow streams. The three RGB streams represent the pyramid appearance cues whereas the three optical flow streams represent the pyramid motion cues. The last CNN pooling layer outputs are then processed by encoding to generate final video descriptors, which are able to recognize a variety of fine-grained human actions.

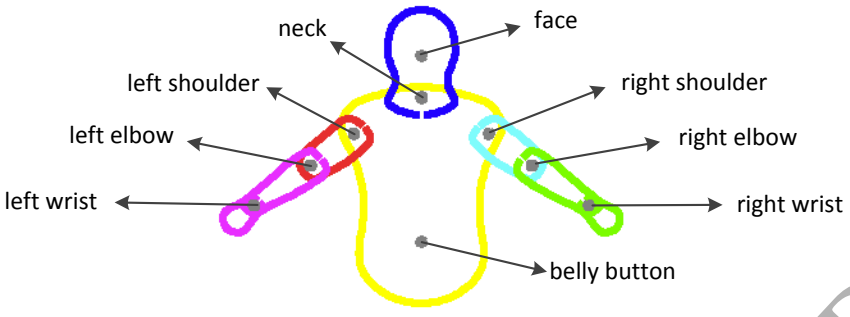


Figure 2: A sample DS puppet model for the human upper body. Colored wireframes represent different body parts and grey points represent the corresponding keypoints.

3. Method

The procedure and pipeline of our proposed method is illustrated in Fig. 1. In this section we introduce them in detail.

3.1. Human pose estimation in video sequence

We build on our recently published state-of-the-art human pose tracker [17] to track human poses, yielding information-rich features which can then be fed into the CNN architecture. The utilized DS puppet model [45] (see Fig. 2) is a part-based probabilistic model, which represents human body parts as their natural shapes, connected in a kinematic chain. This model is learned from training contours derived from SCAPE [46], which is a 3D model for articulated human shape. The variability of the model parts is gained through Principle Component Analysis (PCA). The DS puppet model and the corresponding keypoints are illustrated in Fig. 2.

Human body pose candidates are initialized by utilizing the method of flowing DS puppets [47], which proposes DS pose candidates through the entire video. With image evidences from adjacent frames propagated forwards and backwards over time, each frame obtains much richer cues for consistent pose estimation.

In this case, for each frame t in a video sequence $\{t|t \in [1, 2, ..., T]\}$, we have body poses generated for the frame t as well as body poses propagated from frame $t-1$ and $t+1$. The body pose is evaluated by a contour-based term $p_{ct}(I_t|\mathbf{x}_t)$, a color based term $p_{cl}(I_t|\mathbf{x}_t)$, and a hand likelihood term $p_h(I_t|\mathbf{x}_t)$ [17], as shown in Eq. (1):

$$p(I_t|\mathbf{x}_t) = \lambda_{ct} p_{ct}(I_t|\mathbf{x}_t) + \lambda_{cl} p_{cl}(I_t|\mathbf{x}_t) + \lambda_h p_h(I_t|\mathbf{x}_t). \quad (1)$$

where, I_t means the image of frame t in the video sequence, and $\mathbf{x}_t = [\mathbf{k}_t, s_t]$ represents the vector of DS model variables. $\mathbf{k}_t$ represents body pose keypoints location vector, which contains 2D location information

of nine elements, *i.e.* belly button, face, neck, left/right shoulders, left/right elbows as well as left/right wrists. s_t means the scale which used to fit human pose models into certain images of various size. λ_{ct}, λ_{cl} and λ_h are fixed coefficients whose values and the selection criteria are shown in Table 1.

Using the cost function given by Eq. (1), we are able to select one pose candidate with the highest score for each frame, but the consistency throughout the entire video sequence cannot be ensured. Besides, in our case *i.e.* for the problem of human action recognition in videos, we only need to use the human pose estimation outcome to extract various types of image patches which contain multiple action information. That is to say, in our specific work, coarse but consistent human pose results are enough, and there are no needs to seek the exact positions of each key point of the human body.

In order to assure the consistency of human pose estimation, we add an additional term called *consistency penalty term* to the cost function Eq. 1. The penalty term, denoted by $p_{cs}(I_{t-1}, I_t, I_{t+1}|\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1})$, helps filtering inconsistent pose candidates in the video sequence.

In frame t $(1 < t < T)$, the estimated pose candidate variable $\mathbf{x}_t$ is then propagated to frame $t+1$ using the optical flow affine matrix to get a pose candidate variable $\widehat{\mathbf{x}}_{t+1} = [\widehat{\mathbf{k}}_{t+1}, \widehat{s}_{t+1}]$. Similarly, $\mathbf{x}_t$ is also propagated to frame $t-1$ to get $\widehat{\mathbf{x}}_{t-1} = [\widehat{\mathbf{k}}_{t-1}, \widehat{s}_{t-1}]$. As a result, we are able to calculate two distance terms:

$$d_{t,t-1} = \sum_{i=1}^{N} |\widehat{\mathbf{k}}_{t-1}(i) - \mathbf{k}_{t-1}(i)|; \quad d_{t,t+1} = \sum_{i=1}^{N} |\widehat{\mathbf{k}}_{t+1}(i) - \mathbf{k}_{t+1}(i)|. \tag{2}$$

At an initial attempt, we chose the coarse consistency penalty term to be the average of two distance terms. However, we found that this would lead to over-penalty when the optical flow matrix was not reliable. To resolve this problem, we compute a summarized distance d_c between neck and face, neck and left shoulder, and neck and right shoulder. By selecting $d_c/2$ as a threshold, we define the consistency penalty term to be:

$$p_{cs}(I_{t-1}, I_t, I_{t+1}|\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}) = \begin{cases} w \times (d_{t,t-1} + d_{t,t+1})/2, & if < d_c/2 \\ d_c/2 \quad , & otherwise \end{cases} \tag{3}$$

where, w is a weight coefficient. In our implementation, we set this value to be $w = 2$ found by trial and error.

For the frame $t = 1$, the consistency penalty term is defined as:

$$p_{cs}(I_{t-1}, I_t, I_{t+1}|\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}) = p_{cs}(I_t, I_{t+1}|\mathbf{x}_t, \mathbf{x}_{t+1}) = \begin{cases} w \times d_{t,t+1}, & if < d_c/2 \\ d_c/2 \ , & otherwise \end{cases} \tag{4}$$

while for the frame $t = T$, the consistency term is defined as:

$$p_{cs}(I_{t-1}, I_t, I_{t+1}|\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}) = p_{cs}(I_{t-1}, I_t|\mathbf{x}_{t-1}, \mathbf{x}_t) = \begin{cases} w \times d_{t,t-1}, & if < d_c/2 \\ d_c/2 \ , & otherwise \end{cases} \tag{5}$$

Then the cost function for evaluating consistent human poses in video sequences for each frame is defined as:

$$s = \lambda_p p(I_t|\mathbf{x}_t) + \lambda_{cs} p_{cs}(I_{t-1}, I_t, I_{t+1}|\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}) \tag{6}$$

where, $p(I_t|\mathbf{x}_t)$ is the cost function for human part pose candidates in each image frame as illustrated in Eq. (1). λ_p and λ_{cs} are fixed coefficients, which are described in Table 1.

The parameter values used to test the method and their corresponding selection criteria are summarized in Table 1 based on their applications and magnitudes. The values of the parameters reported in Table 1 are fixed for all our experiments *i.e.* for all video sequences of various datasets.

Table 1: List of the parameters used in the experiments and corresponding selection criteria.

Equation	Coefficients	Selection
Cost function for each image Eq. (1)	$\lambda_{ct} = 4$, $\lambda_{cl} = 1$, $\lambda_h = 1$	$0 < \lambda_h \leq \lambda_{cl} < \lambda_{ct}$
Cost function for video sequence Eq. (6)	λ_p=1, $\lambda_{cs} = -3$	$\lambda_{cs} < 0 < \lambda_p$

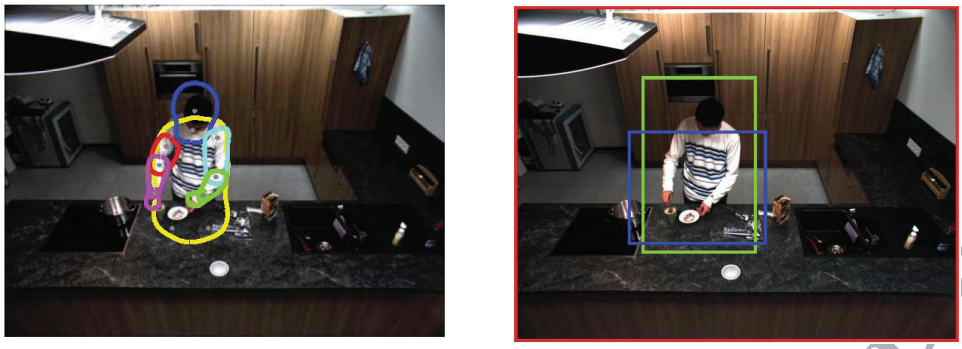
3.2. Obtaining Contextual Cues Regions

Information presented by a video sequence is often divided into spatial and temporal information. The spatial information exists in each individual frame, *i.e.* scenes and objects; while the temporal information lies within the motion between adjacent frames, *i.e.* the movement of observers or cameras and the motion of objects in the scene. In order to obtain rich temporal information, in addition to the original RGB video frame images, we also calculate the optical flow images for each pair of adjacent frames.

3.2.1. Appearance cues

For any given video frame, we first extract the pose of the human actor using our method described in Section 3.1, for example, as shown in Fig. 3(a). Next, for the given frame, we use the obtained pose information to extract three different area patches: full image area (red), human area (green) and operation area (blue), as shown in Fig. 3(b). Full image patches are obtained by resizing the original video frames to be 224×224.

Using the human pose estimation method illustrated in Section 3.1, we are able to obtain points of entire human pose contours. The top left and bottom right points of the counters are calculated and are represented



(a) Human pose estimation result (b) Patches extraction

Figure 3: Crop the raw video frame image utilizing the pose estimation result to get different types of patches. (a) the human estimation result for each video frame. (b) different color boxes represents the crop edges: red box is for the full image patch, green box represents the body patch, while blue box means the operation patch.

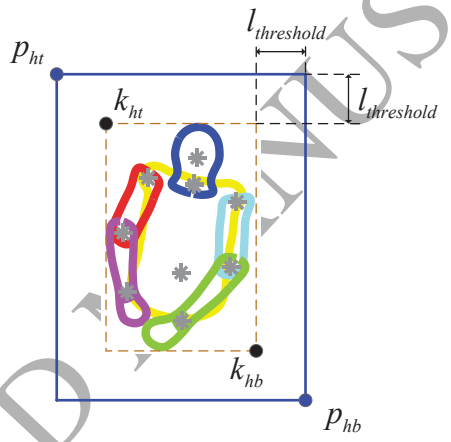


Figure 4: Calculation and selection of the body area.

as k_{ht} and k_{hb} respectively, as illustrated in Fig. 4. By assuming the size of the original video image frame t as $[a \times b]$, we define and set a patch margin threshold $l_{threshold}$ as shown in Eq. (7).

$$l_{threshold} = \frac{min(a, b)}{10} \times s_t \tag{7}$$

where, s_t represents the scale used to fit human pose models into certain images of various size (see explanation below Eq. (1)).

Using k_{ht}, k_{hb}, and $l_{threshold}$, we are able to calculate the top left point p_{ht} and the bottom right point p_{hb} of the human body area, as defined in Eq. (8) and shown in Fig. 4. The corresponding human body

patch is obtained by resizing the region among p_{ht} and p_{hb} to be 224×224, as shown in Fig. 6(b).

$$\begin{cases} p_{ht}(x) = k_{ht}(x) - l_{threshold}, \\ p_{ht}(y) = k_{ht}(y) - l_{threshold}, \end{cases} \quad \begin{cases} p_{hb}(x) = k_{hb}(x) + l_{threshold}, \\ p_{hb}(y) = k_{hb}(y) + l_{threshold}. \end{cases} \tag{8}$$

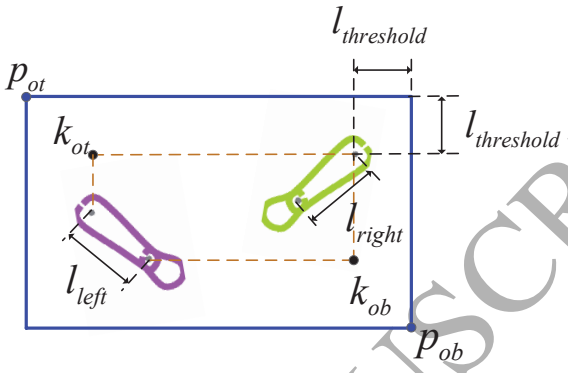


Figure 5: Calculation and selection of the operation area.

The calculation of operation patch is shown in Fig. 5. Operation patch represents the visual details around human actor's lower arms. We first obtain the top left k_{ot} and bottom right k_{ob} points of left/right elbows and left/right wrists. Then, we calculate the lengths l_{left} and l_{right}, which represent the length of left and right lower arms, respectively. Next, the top left (p_{ot}) and the bottom right (p_{ob}) endpoints of the operation area are computed as shown in Eq. (9).

$$\begin{cases} p_{ot}(x) &= k_{ot}(x) - l_{threshold} - \frac{1}{2} \times l_{max}, \\ p_{ot}(y) &= k_{ot}(y) - l_{threshold} - \frac{1}{2} \times l_{max}, \end{cases} \quad \begin{cases} p_{ob}(x) &= k_{ob}(x) + l_{threshold} + \frac{1}{2} \times l_{max}, \\ p_{ob}(y) &= k_{ob}(y) + l_{threshold} + \frac{1}{2} \times l_{max}, \end{cases} \tag{9}$$

where, $l_{max} = max(l_{left}, l_{right})$ represents the maximum length of lower arms.

The operation area is the region from p_{ot} to p_{ob}. We resize this region to be 224×224, and get the operation patch as shown in Fig. 6(c).

3.2.2. Motion cues

As well as obtaining appearance information from the extracted three image patches, we calculate optical flow images between adjacent video frames using the method proposed by Brox *et al.* [48], Fig. 7 shows the magnitude image of the two-dimension optical flow image for two adjacent images calculated using the

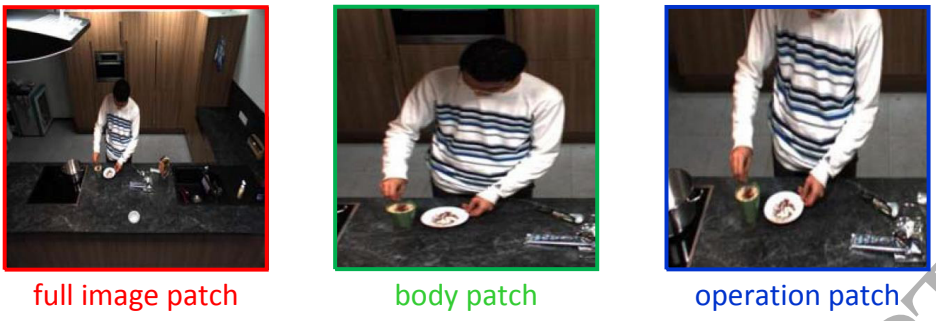


Figure 6: Three RGB patches for each frame of fine-grained action video sequence.

method of Brox *et al.* [48]. The optical flow image for each frame $t-1$ and t is represented as $\hat{U}_t$:

$$\hat{U}_t = \begin{cases} f_{flow}(I_t, I_{t+1}), \ if \ t = 1 \\ f_{flow}(I_{t-1}, I_t), \ otherwise \end{cases} \tag{10}$$

where, $f_{flow}(\cdot)$ represents the method of calculating the optical flow matrix for two images proposed by [48].

According to the requirement of flow-CNN (described in Section 3.3), patches should be of three channels, so we need to transfer every $[a \times b \times 2]$ dimension optical flow image $\hat{U}_t$ which is calculated by Eq. (10) into an $[a \times b \times 3]$ matrix. This has been performed in the following steps. First, using Eq. (11) we calculate $|U(x,y)|$, which is the magnitude of $\hat{U}_t$. Later, this magnitude is processed using the linear transformation shown in Eq. (12). The resultant matrix of this step serves as the additional third channel of the new optical flow image U_t. Next, the first two channels of U_t are calculated using Eq. (13). Finally, all three channels are concatenated to form the new optical flow image U_t. Fig. 8 shows the computed three channels of the optical flow image.

$$|U(x,y)| = \sqrt{\widehat{U}_t(x,y,1)^2 + \widehat{U}_t(x,y,2)^2} \tag{11}$$

$$U_t(x,y,3) = 16 \times |U(x,y)| \tag{12}$$

$$U_t(x,y,c) = 16 \times \widehat{U}_t(x,y,c) + 128, \quad \text{where } c = 1,2 \tag{13}$$

Similar to the RGB patches in Fig. 6, we also extract three patches for each frame, *i.e.* full image patch, body patch and operation patch, as shown in Fig. 9.

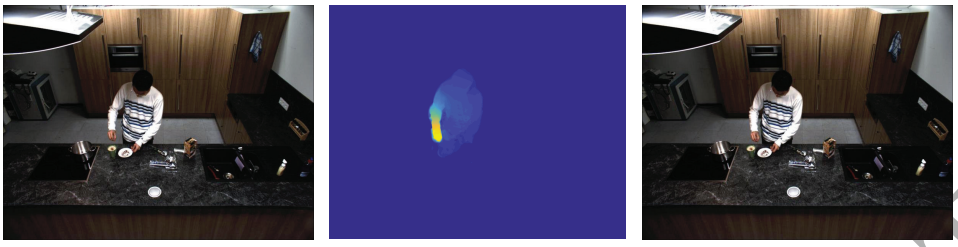


Figure 7: Calculation of the optical flow image for adjacent two video frames. (a) The image of frame $t-1$; (b) The optical flow magnitude between the frame $t-1$ and t; and (c) The image of frame t.

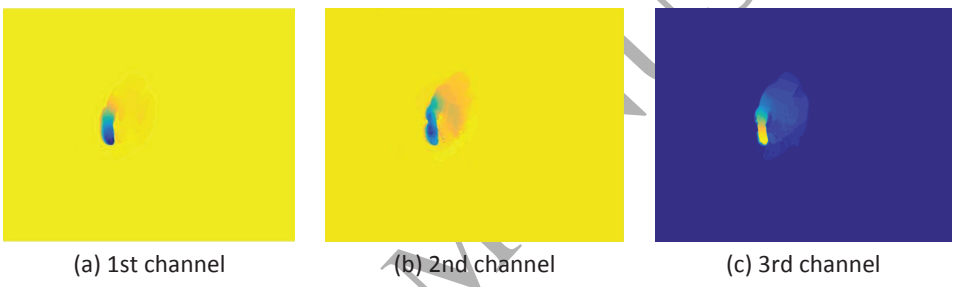


Figure 8: Three channels of optical flow patches for each frame of fine-grained action video sequence.

Figure 9: Three optical flow patches for each frame of fine-grained action video sequence.

3.3. Convolutional Neural Network Structures and Action Descriptor Construction

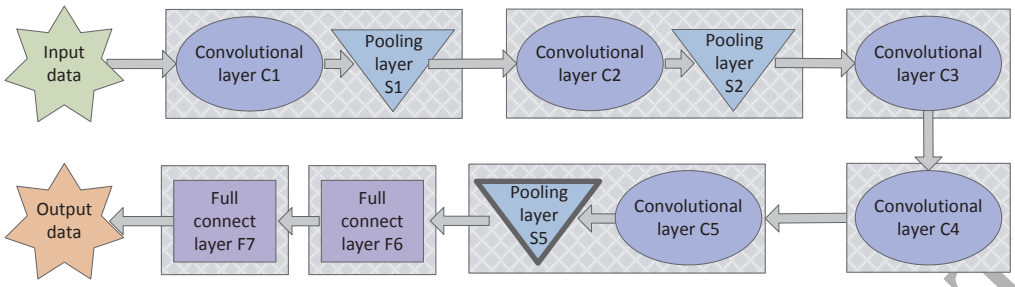
Considering that the properties for RGB images and flow images are different, we use similar but different CNN networks for RGB and flow patches separately. For simplicity, we call them RGB-CNN and flow-CNN respectively. Each of these networks have five convolutional layers and three pooling layers, as shown in Table 2. In Table 2, $n_1 \times k_1 \times k_1$ for convolutional layers mean using n_1 numbers of $k_1 \times k_1$ kernels, and $k_2 \times k_2$ for pooling layers mean using $k_2 \times k_2$ kernels. Besides, in Table 2, "str" means stride, and "pad" means padding. To make our proposed method more general, unlike some other related works which train data and parameters for each dataset specifically and individually, we use the same pre-trained CNN configurations for all the video sequences of various datasets. In our method, the data and parameters of the RGB-CNN architecture are pre-trained by Chatfield and Simonyan [49] using ILSVRC-2012 dataset [50]; while the data and parameters of the flow-CNN are trained by Gkioxari and Malik [42] using UCF101 dataset [51].

Table 2: CNN structures

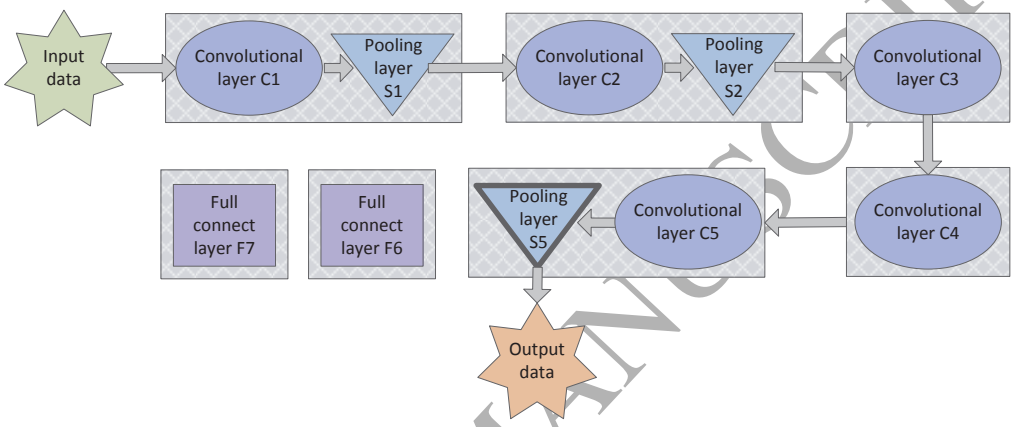
Layer	$conv_1$	$pool_1$	$conv_2$	$pool_2$	$conv_3$	$conv_4$	$conv_5$	$pool_5$	fc_6	fc_7
RGB-CNN	64x11x11 str 4 pad 0	2x2	256x5x5 str 1 pad 2	2x2	256x3x3 str 1 pad 1	256x3x3 str 1 pad 1	256x3x3 str 1 pad 1	2x2	4096	4096
flow-CNN	96x7x7 str 2	3x3	384x5x5 str 2	3x3	512x3x3 str 1	512x3x3 str 1	384x3x3 str 1	3x3	4096	4096

As depicted in Table 2, $pool_5$ refers to the features of the last pooling layer, while fc_6 and fc_7 represent the features of first and second fully connected layers, respectively. Most current related works, for example, Girshick *et al.* [15] and Cheron *et al.* [10], use features of fc_6 or fc_7 to construct image descriptors, as illustrated in Fig. 10(a). However, other works claim that $pool_5$ contains more spatial information and could provide more distinguishable descriptors [52]. In our proposed work, we use the features of $pool_5$ to construct image descriptors, as illustrated in Fig. 10(b). Consider RGB-CNN for instance, the $pool_5$ layer has $6 \times 6 \times 256$ features, which corresponds to a vector of 9216 dimension, while fc_6 and fc_7 features are of 4096 dimension. In order to reduce the computational cost, we obtain reduced dimension by encoding. The most common used encoding methods are Fisher vector encoding [53] and vector of locally aggregated descriptors (VLAD) [54]. Xu *et al.* [52] analyzed the discriminating ability of Fisher and VLAD encoding methods, and found that VLAD is more feasible for CNN descriptors.

Assuming that the size of features in $pool_5$ layer is $a \times a \times b$, we reshape them into a^2 features, each of which is of b dimension, and we denote the new reshaped features as $\{x_1, x_2, ..., x_{a^2}\}$. For entire video with T number of frames, a^2T features are obtained, which are denoted as $\{x_1, x_2, ..., x_{Ta^2}\}$. For each fine-grained action video, we extract six image patch sequences as shown in Fig. 6 and Fig. 9. We feed each image patch into corresponding CNN structure, and consequently we get a^2T features of b dimension for each patch sequence. Imaging that how huge amount the total features for fine-grained action sequence should be. In



(a) Common usage of CNN architecture, which extract data from fc_7 layer.



(b) Our proposed usage of CNN architecture, which extract data from $pool_5$ layer.

Figure 10: Emphasize CNN architecture used in our proposed system.

order to extract efficient information from these CNN features, we propose an encoding method. Note that different sequences focus on different information, for instance, RGB upper body patches focus on human pose spatial information, optical flow operation patches focus on hands and object motion information, etc. In this case, we handle each type of patch sequence separately. In the following description, we use RGB operation patch sequence as an example, but the handling method is the same for the other five sequence.

We assume that the number of fine-grained action video in the training dataset is N_{train}, and the length of all these videos are $\{T_1, T_2, ..., T_{N_{train}}\}$ separately. For RGB operation patch sequence, the number of b dimension features are $(T_1 + T_2 + ... + T_{N_{train}}) \times a^2$. We use k-means [55] to cluster these features, see Algorithm 1.

Based on Algorithm 1, we obtain K cluster centers for all RGB operation patch sequences of training dataset, denoted as $\{c_1, c_2, ..., c_K\}$. Then for each sequence of testing dataset, we cluster the corresponding a^2T features of b dimension, and then calculate the nearest cluster center for each feature, and store the

Algorithm 1 Use k-means to cluster $pool_5$ features.

Select $K = 128$ cluster centers of dimension b;
while centers not stable **do**
  for $i = 1, 2, ..., (T_1 + T_2 + ... + T_{N_{train}})$ **do**
    for $j = 1, 2, ..., K$ **do**
      Calculate distance between ith vector and jth center, denoted as $d_{i,j}$;
    end for
    Calculate $k_i = \underset{j=1,2,...,K}{argmin}\ d_{i,j}$ as the index of cluster center for ith vector;
  end for
  for $j = 1, 2, ..., K$ **do**
    Calculate the mean location of vectors belong to the jth cluster as the updated jth center;
  end for
end while
Output the updated K centers.

index as $NN(x_i)$, as follows:

$$NN(x_i) = \underset{j=1,2,...,K}{argmin}\ ||x_i - c_j||. \tag{14}$$

Focusing on each cluster center c_j, we calculate distance between c_j and x_i which satisfies $NN(x_i) = j$, and summarize the distance as u_j:

$$u_j = \sum_{x_i : NN(x_i) = j} (x_i - c_j). \tag{15}$$

Now, the VLAD encoding vector is obtained by concatenating all the u_j of k-means cluster centers, which is denoted as $f = [u_1, u_2, ..., u_K]$. The dimension of feature f is $b \cdot K$.

We use three RGB patches and three flow patches for each frame, and we suppose that the size of $pool_5$ features of the RGB-CNN and the flow-CNN are $a_1 \times a_1 \times b_1$ and $a_2 \times a_2 \times b_2$, respectively. We denote VLAD descriptor of RGB full image patch, body patch, and operation patch as f_1, f_2, and f_3 separately; we denote VLAD descriptor of optical flow image patch, body patch, and operation patch as f_4, f_5, and f_6 respectively. Finally, we concatenate the obtained VLAD descriptors, to generate a fine-grained action descriptor for video as follows:

$$f_{action} = [f_1, f_2, f_3, f_4, f_5, f_6]. \tag{16}$$

The obtained fine-grained action descriptor has $3(b_1 + b_2)K$ dimensions. In our implementation, we calculate f_{action} of training dataset, and then use these video descriptors to train SVM (Support Vector Machines) classifier [56]. The obtained SVM classifier is able to recognize fine-grained actions from different videos.

4. Experiments

4.1. Dataset

Two publicly available human action datasets have been used for evaluation experiments. The first one is sub-JHMDB dataset [57], as shown in Fig. 11, which is proposed by Jhuang *et al.* contains 316 videos of twelve different actions, which are catch, climb stairs, golf, jump, kick ball, pick, pull-up, push, run, shoot ball, swing baseball and walk. This dataset is provided with three kinds of training/testing split modes. The other dataset is MPII Cooking dataset [7], as shown in Fig. 12, which contains 65 different cooking activities such as cut slices, pour spice, wash objects, etc., recorded from 12 participants. In total there are 44 videos with a total length of more than 8 hours. This dataset is provided with seven kinds of training/testing split modes. In our experiments, we test on each split mode of each dataset, and calculate the average accuracy among all splits.

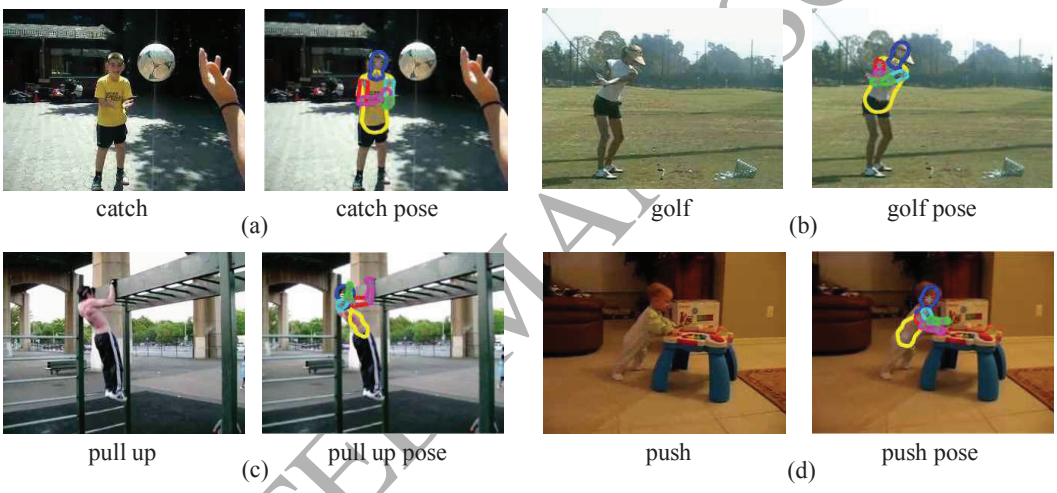


Figure 11: Sample frames of sub-JHMDB dataset. We show each frame twice: firstly the original video image; secondly the image overlaid with the estimated human pose which is used to extract human region patches.

From Fig. 12 it is seen that with MPII Cooking dataset, most human action information is contained in the upper body, while with sub-JHMDB dataset (see Fig. 11), the action information is encoded in the entire human body pose. In this case, for the computation of body path on the sub-JHMDB dataset, we obtain the person's torso bottom position k_{belly} estimated in Section 3.1, and mirror the human upper body (obtained in Section 3.2) vertically using the center of k_{belly}, as shown in Fig. 13(b). The region within the upper body rectangle patch and the vertically mirrored patch is denoted as body patch for the calculation on the sub-JHMDB dataset, as shown in Fig. 13(c).

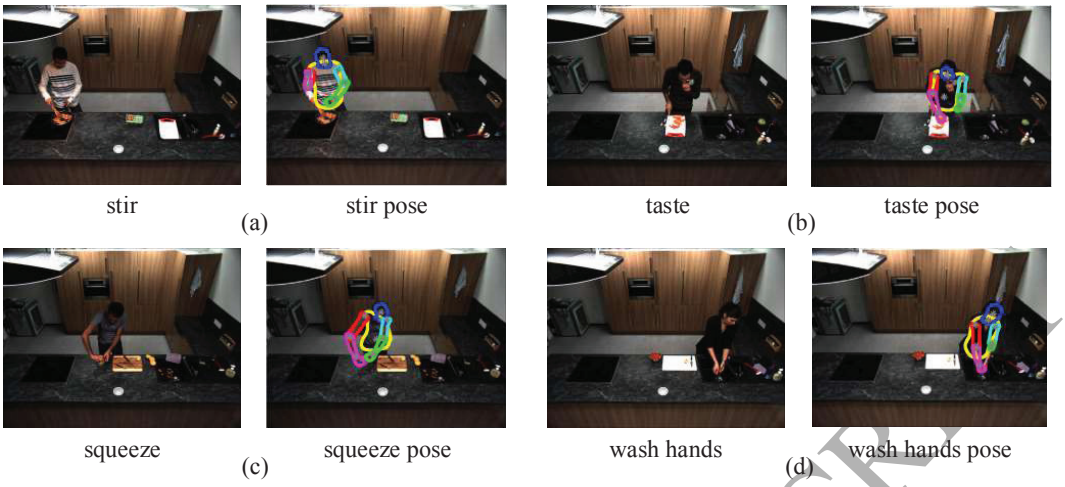


Figure 12: Sample frames of MPII Cooking dataset. We show each frame twice, the first one is the original video frame from dataset, the second one is overlaid with the estimated human pose which is used to extract human region patches.

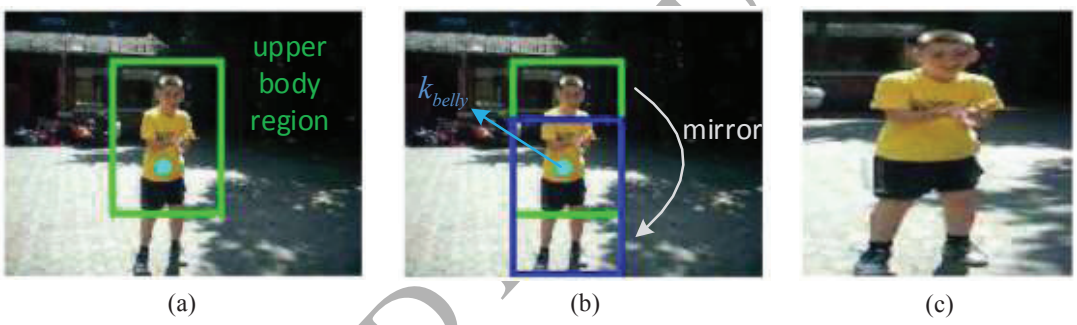


Figure 13: Obtaining body patches for the sub-JHMDB dataset. (a) Human upper body patch (green); (b) Mirrored green patch vertically with the center of belly button to get blue patch; (c) The obtained whole body patch.

4.2. Preliminaries

In video sequences, human movement is typically continuous. However, through observation we found that some adjacent frames are similar while others have significantly different human position and pose. That is to say, different frames contribute different amounts of information for human fine-grained action recognition in videos. In this case, the calculation of similar frames in video may lead to redundant information and increase the amount of calculation.

The problem described above is not serious in short sequences. However, the calculation of k-means centers for long videos of the MPII Cooking dataset as expounded in Algorithm 1 causes very large cost of computation time and computation space. To solve this problem, when calculating k-means centers for $pool_5$ layer (Algorithm 1), we calculate and select key frames of each video sequence instead of using all the

frames. This has been accomplished as follows and illustrated in Fig. 14.

For every video (whose length is denoted to be T), we use the features of fc_7 layer from the CNN structure of the RGB full image patches to form a $T \times 4096$ matrix. The columns whose elements are all zeros are removed, and then we select the largest elements (denoted as "m") from the remaining matrix (denoted as C). We transfer every element of C into binary sequence with the length of $\log_2 m$. In this case, the obtained new matrix is denoted by $\hat{C}$, as shown in Fig. 14.

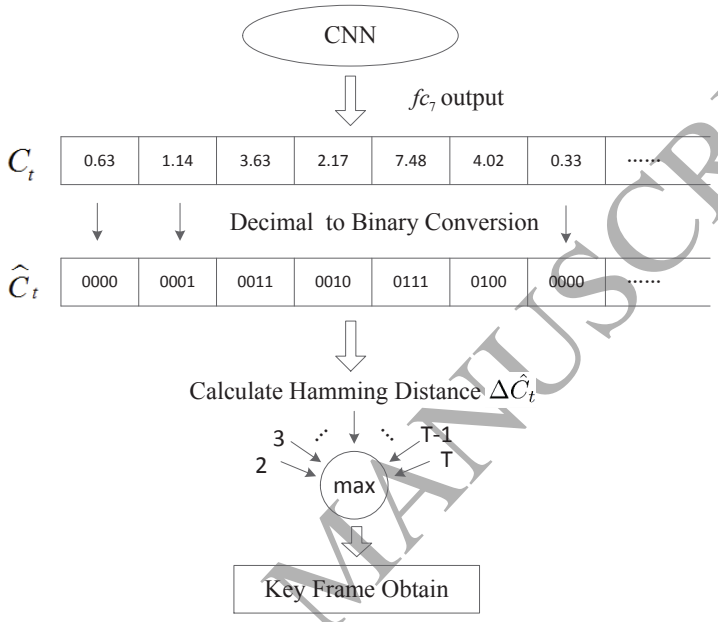


Figure 14: Calculate and select key frames from video sequence using CNN fc_7 layer output.

For $t = 2, 3, ..., T$, we calculate the *hamming distance* between $\hat{C}_{t-1}$ and $\hat{C}_t$, which is denoted as $\Delta\hat{C}_t$, see Eq. (17).

$$\Delta\hat{C}_t = hamming(\hat{C}_t - \hat{C}_{t-1}), \quad t = 2, 3, ..., T \tag{17}$$

Ranking all the frames according to descending order of the hamming distance $\Delta\hat{C}_t$, we select k frames with the largest hamming distance, which are called key frames. These key frames are then used to calculate the k-means centers for the VLAD encoding.

The number of k-means cluster centers K (see Algorithm 1) should always be greater than the total number of actions. This can be justified by analyzing the clustering performance with different values of K. For instance, consider the calculation of cluster centers (as described in Algorithm 1) for the RGB full image patch sequence of the MPII Cooking dataset, which has 65 actions. For a range of $K = [32, 256]$, the average value of cluster radiuses (average cluster radius, ACR) are calculated. A cluster radius is the largest distance between each point and the corresponding cluster center, and can be used for evaluating the

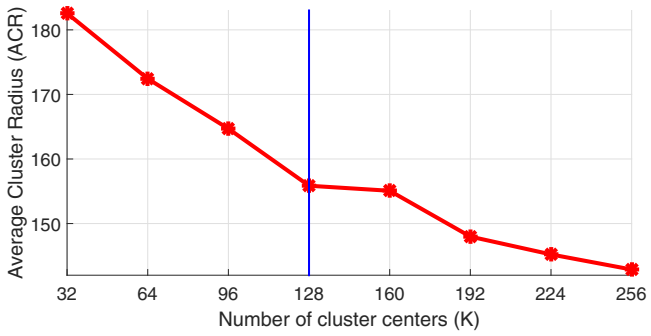


Figure 15: Relationship between the number of cluster centers K and the average cluster radius. Smaller the ACR better the clustering performance.

clustering performance *i.e.,* smaller the ACR better the performance. From Fig. 15, it can be seen that the performance improves with the number of cluster centers (larger K). Although the performance is improved, with higher values of K, overall computation time also increases. Hence, an appropriate balance between cluster performance and calculation time should be maintained while selecting K. From Fig. 15 we can see that the performance rapidly improved until $K = 128$ and progressed gradually afterwards. Therefore, in our experiments, we choose the number of cluster centers to be $K = 128$ (step 1 of Algorithm 1).

Also it is worth mentioning that all the experiments are conducted on a lab computer running Ubuntu 14.04 with 2.80GHz Intel Core i7 CPU and 16 GB of RAM. We have used Matlab for implementation purposes.

4.3. Action classification performance

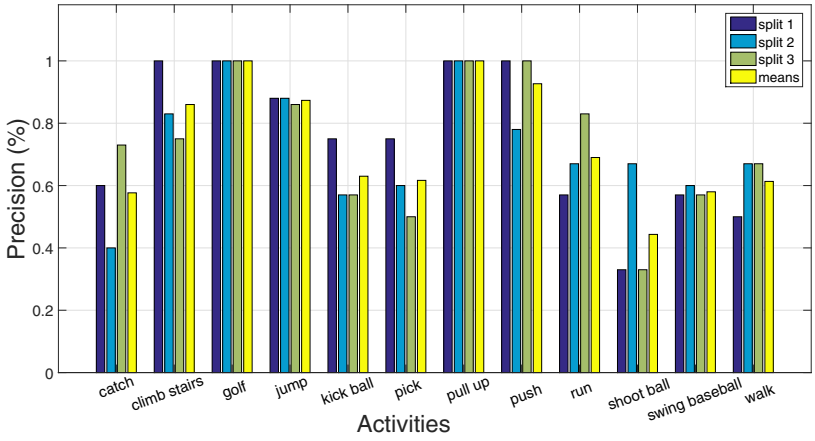
Our proposed action recognition system is able to classify fine-grained actions, but also works well for general physical actions. For the reason to verify that our proposed action recognition method is not limited to fine-grained actions, we use a very commonly used public human action dataset to test the system's performance. In this section, we use sub-JHMDB dataset to test the precision and recall performance of our proposed system. The definition of evaluation criterion "precision" and "recall" are described in Eq. (18) and Eq. (19) separately.

$$precision = \frac{count((recognized\ as\ action\ A) \cup (action\ A\ in\ dataset))}{count(recognized\ as\ action\ A)} \tag{18}$$

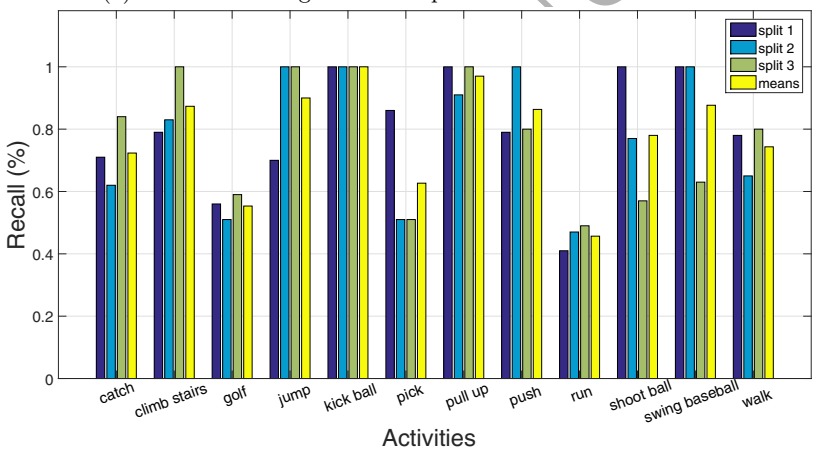
$$recall = \frac{count((recognized\ as\ action\ A) \cup (action\ A\ in\ dataset))}{count(action\ A\ in\ dataset)} \tag{19}$$

The detailed per-class human actions classification results are shown in Fig. 16. From Fig. 16(a) we can see that the action classification precision of actions such as golf, pull up, push, jump and climb stairs are very

high, with mean precision of 100%, 100%, 92% 87% and 86% respectively. The mean action classification precision of catch, kick ball, pick, run, swing baseball and walk are 57%, 63%, 61%, 69%, 58%, 61% and 73% respectively, which is somewhat less good, but is still highly competitive compared with other state-of-the-art methods. The mean precision of shoot ball is the lowest, at 44%.



(a) Precision bar figure for all splits of sub-JHMDB dataset.



(b) Recall bar figure for all splits of sub-JHMDB dataset.

Figure 16: Performance on the dataset of sub-JHMDB.

Fig. 16(b) reveals the recall of each training/testing split for sub-JHMDB dataset. As presented in Fig. 16(b), the mean recall of actions such as climb stairs, jump, kick ball, pull up, push and swing baseball are more than 80%, while catch, pick, shoot ball and walk are between 60% - 80%. The mean recall of golf and run are 55% and 46% respectively. From the performance revealed in Fig. 16, it is obvious that our proposed method performs well for common physical action recognition problem.

4.4. Comparison with state-of-the-art methods

In this section, we present the experimental results obtained by validating our proposed human actions classification method against various other state-of-the-art methods proposed by Wang *et al.* [26], Gkioxari *et al.* [42], Peng *et al.* [24] and Cheron *et al.* [10] on the sub-JHMDB dataset (Table 3), and against the methods of Ni *et al.* [58] and Rohrbach *et al.* [7] on MPII Cooking dataset (Table 4). The reason that we use different comparison methods for different testing dataset is that the compared methods are the state-of-the-art methods which have been published on each dataset respectively. I.e. we compare against the best methods on each dataset for which performance data is publicly available.

Table 3: Performance on sub-JHMDB dataset compared with related state-of-the-art methods.

Method	Accuracy
Wang *et al.* [26]	56.6%
Gkioxari *et al.* [42]	62.5%
Peng *et al.* [24]	69.3%
Cheron *et al.* [10]	72.2%
Ours	76.9%

Table 3 shows our proposed system performance on sub-JHMDB dataset compared with related state-of-the-art methods. The "accuracy" in Table 3 means the proportion of correct recognized in all the videos of test dataset. As reported by Wang *et al.* [26], they proposed a method which describe videos by sampling dense points from each frame and tracking them based on displacement information from a dense optical flow field. Gkioxari and Malik [42] proposed a method which selects image regions containing salient motion and use spatial and temporal information to build action representations. These two methods extract image regions utilizing motion information, while we use our human pose tracking method to obtain consecutive regions which contain human action information. Table 3 reveals that on the sub-JHMDB dataset the accuracy of our proposed method is **20.3%** higher than the method of Wang *et al.* [26], and **14.4%** higher than the method of Gkioxari and Malik [42].

Peng *et al.* [24] proposed *stacked Fisher vectors* to represent action features. This method uses Fisher vectors to encode local features extracted from the densely sampled sub-volumes in the first layer, and compress the sub-volumes Fisher vectors as well as encodes them again with Fisher vectors. Cheron *et al.* [10] proposed a method which aggregates motion and appearance information along tracks of human body parts through CNN structures. Instead of using stacked Fisher vectors or soft-max CNN features, we proposed a VLAD encoding method using $pool_5$ layer features. From Table 3 we can see that the accuracy of our proposed method is **7.6%** higher than the method by Peng *et al.* [24], and **4.7%** higher than the method of Cheron *et al.* [10].

Table 4 shows our proposed system performance on MPII Cooking dataset compared with related state-

Table 4: Performance on MPII Cooking dataset compared with related state-of-the-art methods.

Method	Precision	Recall	Average Precision (AP)
Ni *et al.* [58]	28.6%	48.2%	54.3%
Rohrbach *et al.* [7]	50.4%	45.1%	57.9%
Ours	57.7%	56.4%	70.3%

of-the-art methods. The "Average Precision (AP)" in Table 4 means the performance obtained by drawing a precision-recall curve based on the SVM classification score and calculating the area under the curve (AUC).

Ni *et al.* [58] proposed a strategy to detect and define the interaction between hands and objects. This strategy infers coarse interaction status and uses the obtained information to get compact action feature. Instead of focusing only on hand and object to construct action features, our proposed method extracts image patches according to the estimated human pose and uses six different patch sequences containing spatial and temporal information. From Table 4 it can be seen that the precision and recall of our proposed method is **29.1%** and **8.2%** higher than the method of Ni *et al.* , and the average precision of our method is **16%** higher than Ni *et al.* method.

Rohrbach *et al.* [7] proposed a method which estimates human pose joint points among frame sequence, and compute a separate codebook for each distinct sub-feature (*i.e.* velocity, acceleration, exponential bands etc.), then extracts histograms of oriented gradients (HOG), histograms of optical flow (HOF), and motion boundary histograms (MBH) around densely sampled points. With all these features and methods, the average precision of [7] is 57.9%, and their precision and recall are 50.4% and 55.1% respectively. Instead of relying on hand-crafted features, we use the $pool_5$ layer features of CNN structure, which are more intelligent and distinguishable. The results are summarized in Table 4. It can be seen that our precision, recall and average precision are **7.3%**, **11.3%** and **12.4%** higher than Rohrbach *et al.* .

4.5. Contribution of each system component to the overall performance

In this section we examine the contribution of each part of our proposed method to overall performance on MPII Cooking dataset. With our method, firstly we proposed a pose tracking and estimation schema for constructing human action region patches. To verify the efficiency of pose estimation method, we compute the CNN features based only on RGB and flow full-image patches. The obtained results are shown in the **first row** of Table 5. From these, we notice that the human action recognition precision and recall, without using our proposed body patches and operation patches approach, are 37.3% and 43.5%, which are lower than our full method by 20% and 12.9% respectively. The average precision by using only full image patch is 12.9% lower than our patch-based method. This is because the background of full images contains large amounts of redundant information that is less useful than foreground patches. In contrast, our human pose estimation method is able to extract and enhance foreground patches and make better use of the most

Table 5: Contribution of each system component to the overall performance on MPII Cooking dataset.

No.	Method	Precision	Recall	Average Precision (AP)
1	fp(both)+$pool_5$+VLAD	37.7%	43.5%	57.4%
2	fp(both)+bp(both)+$pool_5$+VLAD	41.8%	50.9%	64.6%
3	fp(RGB)+bp(RGB)+op(RGB)+$pool_5$+VLAD	48.4%	46.5%	67.8%
4	fp(both)+bp(both)+op(both)+fc_7+max	42.1%	53.0%	62.1%
5	fp(both)+bp(both)+op(both)+fc_7+average	30.8%	28.7%	36.4%
6	fp(both)+bp(both)+op(both)+fc_7+VLAD	47.1%	54.9%	64.2%
7	fp(both)+bp(both)+op(both)+$conv_4$+VLAD	36.2%	42.3%	40.7%
8	fp(both)+bp(both)+op(both)+$pool_5$+VLAD	57.7%	56.4%	70.3%

[1] "fp" means full image patch, "bp" means body patch, and "op" means operation patch.
[2] In the parentheses: "RGB" means using RGB patches (see Fig. 6), while "both" means using both RGB patches and optical flow patches (see Fig. 9).

information-rich parts of each image.

An additional contribution of our work is to define and extract the operation areas for human actions, which are denoted by operation patches as shown in Fig. 3. To verify the contribution of our operation patches to overall system performance, we test the MPII Cooking dataset with and without using our operation patches method. The results are shown in the **second row** of Table 5. These results reveal that, by using our operation patch method, the precision, recall and average precision are increased by 15.9%, 5.5% and 5.7% respectively. The reason is that the operation patch emphasizes the information around hand regions which includes hand pose, object, as well as hands interaction information. Human fine-grained action information mostly exists around lower arms, while common action recognition methods often resize the original images and indirectly reduce effective pixels. Our proposed method overcomes this problem by extracting human operation areas and enhancing the corresponding pixels. This improves the action recognition results.

As mentioned before, RGB images offer appearance cues for human actions, while the calculated optical flow image sequence contains motion cues. In order to verify that the motion information from the optical flow images helps to get better results in recognizing human actions, we test our method with and without optical flow images separately. The results are summarized in the **third row** of Table 5. The results illustrate that without the motion information obtained from optical flow images, precision, recall and average precision would decline by 9.3%, 9.9% and 2.5% respectively (compared with the eighth row of Table 5). The reason for the better performance is that the temporal information contained in optical flow images offers human action motion cues, which contributes to human action recognition problem and improves the system performance.

We use $pool_5$ features with VLAD encoding method instead of the common handling of fc_7 features. In order to justify this, we compare our results with the two most commonly used handling methods for fc_7 features. Specifically, we connect the fc_7 features of every image patch per frame as frame descriptor, and calculate the maximum and average number of each dimension of frame descriptors per video separately

as video descriptors. The obtained video descriptors are then used to train the SVM classifier in order to generate the human action recognition results. The results are shown in the fourth and fifth rows of Table 5. It can be seen from the **fourth row** that for the fc_7+max method, the results are lower by 15.6%, 3.4% and 8.2% on precision, recall and average precision, respectively than our $pool_5$+VLAD method (the eighth row in Table 5). The reason is that the "max" calculation would get rid of some useful information and only keep the extreme values caused by distractions. In this case, the remained extreme value features would not be bale to represent the action properly and lead to failure, especially in the case of strong light or other object distractions.

The **fifth row** of Table 5 illustrates the results of using fc_7+average method. The results in this case are much lower than the previous, *i.e.*, lower by 26.9%, 27.7% and 33.9% on precision, recall and average precision respectively, compared with the proposed $pool_5$+VLAD method. The reason is that the "average" calculation only keeps the most common and average values that are not discriminative. However, different fine-grained actions in MPII Cooking dataset are quite similar and the distinguishable features lie only in some specific frames or locations. As a result, the "average" calculation would make the descriptors of different fine-grained actions similar and lose the discrimination quality, due to which the results become lower. However, the proposed VLAD encoding method (see the eighth row in Table 5) does not have this problem and is able to organize video descriptors more effectively, even though these features are of less dimension.

Furthermore, to demonstrate that $pool_5$ layer output data contains more action information than fc_7 layer, *i.e.*, spatial information, we test our method employing fc_7 layer outputs instead of $pool_5$ layer outputs, leaving the other methods and experimental settings (*e.g.* parameters) unchanged. Specifically, we calculate the k-means centers of 4096 dimensions for fc_7 layer outputs on the training dataset, and use VLAD encoding method to encode fc_7 layer outputs of all the frames for each video sequence into video descriptors. The results are shown in the **sixth row** of Table 5. By comparing these results with our $pool_5$+VLAD (results in eighth row), it is illustrated that using the outputs of $pool_5$ layer, the precision, recall and average precision are 10.6%, 1.5% and 6.1% higher than that of using the outputs of fc_7 layer. The main reason is that $pool_5$ layer features contain more abundant spatial information than fc_7 layer features, so the corresponding data are more effective. From these results, it is clear that the proposed feature handling method achieves higher performance in recognizing fine-grained actions in video sequences.

Moreover, in CNN structures, the aim of convolutional layers is to extract certain features such as edges, angles, curves, or more complex higher-order features. Even though the outputs of earlier layers contain more spatial information, the features are eminently raw and are less distinguishable. In order to justify that using the $pool_5$ layer features make an appropriate balance between distinguishable features and adequate spatial information, we test our method employing the fourth convolutional layer $conv_4$ outputs with VLAD

encoding (see Table 2). The data size of the $conv_4$ layer outputs is similar to that of the $pool_5$ layer (which can be recorded as $a \times a \times b$), therefore the processing procedure is the same. The corresponding experimental results are shown in the **seventh row** of Table 5. By comparing the seventh and eighth rows of Table 5, we found that using $conv_4$ layer data makes the precision, recall and average precision decline by 21.5%, 14.1% and 29.6% respectively than that of using $pool_5$ layer features. Despite of the fact that $conv_4$ layer outputs contain more spatial information, the contained features are hardly distinguishable. These results clearly demonstrates that utilizing $pool_5$ layer outputs makes an appropriate balance between adequate spatial information and distinguishable features, which once again proves the effectiveness of the proposed method in recognizing human actions in videos.

5. Conclusion

We have proposed a novel region sequence based six-stream CNN feature for human action recognition in videos, which combines different scales of image appearance information and video motion information. We built on our recent state-of-the-art method for human pose estimation in video sequences, which localizes the human body parts. As a result, we can then crop different scale human region image patches. This approach uses human body part positions as prior knowledge, and makes better use of spatial image information, which enables fine-grained activities to be distinguished. A variety of different scales of appearance and motion patch sequences are processed in CNN structures to provides richer action information. The encoding of the outputs of the CNN pooling layer gives more effective descriptors. Our method offers a general human action recognition method for videos. Our proposed method outperforms six other state-of-the-art methods, in empirical experiments on two publicly available challenging human action datasets.

References

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International Journal of Computer Vision 88 (2) (2010) 303–338.

[2] K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, in: Advances in Neural Information Processing Systems, 2014, pp. 568–576.

[3] Y. Yang, Y. Li, C. Fermuller, Y. Aloimonos, Robot learning manipulation action plans by" watching" unconstrained videos from the world wide web, in: Advancement of Artificial Intelligence, 2015, pp. 3686–3693.

[4] M. Niitsumag, H. Hashimoto, H. Hashimoto, Spatial memory as an aid system for human activity in intelligent space, IEEE Transactions on Industrial Electronics 54 (2) (2007) 1122–1131.

[5] Z. A. Khan, W. Sohn, Abnormal human activity recognition system based on r-transform and kernel discriminant technique for elderly home care, IEEE Transactions on Consumer Electronics 57 (4) (2011) 1843–1850.

[6] V. Krüger, D. Kragic, A. Ude, C. Geib, The meaning of action: a review on action recognition and mapping, Advanced Robotics 21 (13) (2007) 1473–1501.

[7] M. Rohrbach, S. Amin, M. Andriluka, B. Schiele, A database for fine grained activity detection of cooking activities, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 1194–1201.

[8] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld, Learning realistic human actions from movies, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.

[9] H. Jhuang, T. Serre, L. Wolf, T. Poggio, A biologically inspired system for action recognition, in: IEEE International Conference on Computer Vision (ICCV), 2007, pp. 1–8.

[10] G. Cheron, I. Laptev, C. Schmid, P-cnn: pose-based cnn features for action recognition, in: IEEE International Conference on Computer Vision (ICCV), 2015, pp. 3218–3226.

[11] B. Yao, S. Zhu, Learning deformable action templates from cluttered videos, in: IEEE International Conference on Computer Vision (ICCV), 2009, pp. 1507–1514.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale video classification with convolutional neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1725–1732.

[13] C. Schüldt, I. Laptev, B. Caputo, Recognizing human actions: a local svm approach, in: IEEE 17th International Conference on Pattern Recognition, Vol. 3, 2004, pp. 32–36.

[14] J. Liu, M. S. Jiebo Luo, Recognizing realistic actions from videos in the wild, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 1996–2003.

[15] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580–587.

[16] G. Gkioxari, R. Girshick, J. Malik, Contextual action recognition with r*cnn, in: IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1080–1088.

[17] M. Ma, N. Marturi, Y. Li, R. Stolkin, A. Leonardis, A local-global coupled-layer puppet model for robust online human pose tracking, Computer Vision and Image Understanding 153 (2016) 163–178.

[18] R. Poppe, A survey on vision-based human action recognition, Image and Vision Computing 28 (6) (2010) 976–990.

[19] D. M. Gavrila, The visual analysis of human movement: A survey, Computer Vision and Image Understanding 73 (1) (1999) 82–98.

[20] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[21] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005, pp. 886–893.

[22] R. Chaudhry, Avinash, Ravichandran, G. Hager, R. Vidal, Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 1932–1939.

[23] H. Wang, C. Schmid, Action recognition with improved trajectories, in: IEEE International Conference on Computer Vision (ICCV), 2013, pp. 3551–3558.

[24] X. Peng, C. Zou, Y. Qiao, Q. Peng, Action recognition with stacked fisher vectors, in: European Conference on Computer Vision (ECCV), 2014, pp. 581–595.

[25] A. Iosifidis, A. Tefas, I. Pitas, Discriminant bag of words based representation for human action recognition, Pattern Recognition Letters 49 (2014) 185–192.

[26] H. Wang, A. Klaser, C. Schmid, C. L. Liu, Action recognition by dense trajectories, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 3169–3176.

[27] H. Wang, A. Klaser, C. Schmid, C. L. Liu, Dense trajectories and motion boundary descriptors for action recognition, International Journal of Computer Vision 103 (1) (2013) 60–79.

[28] C. Xu, D. Tao, C. Xu, A survey on multi-view learning, arXiv preprint arXiv:1304.5634.

[29] C. Xu, D. Tao, C. Xu, Multi-view learning with incomplete views, IEEE Transactions on Image Processing 24 (12) (2015) 5812–5825.

[30] J. Li, C. Xu, W. Yang, C. Sun, D. Tao, Discriminative multi-view interactive image re-ranking, IEEE Transactions on Image Processing 26 (7) (2017) 3113–3127. doi:10.1109/TIP.2017.2651379.

[31] M. Bregonzio, T. Xiang, S. Gong, Fusing appearance and distribution information of interest points for action recognition, Pattern Recognition 45 (3) (2012) 1220–1234.

[32] J. C. Niebles, L. Fei-Fei, A hierarchical model of shape and appearance for human action classification, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–8.

[33] S. Ma, J. Zhang, N. Ikizler-Cinbis, S. Sclaroff, Action recognition and localization by hierarchical space-time segments, in: IEEE International Conference on Computer Vision (ICCV), 2013, pp. 2744–2751.

[34] T. Lan, Y. Zhu, A. R. Zamir, S. Savarese, Action recognition by hierarchical mid-level action elements, in: IEEE International Conference on Computer Vision (ICCV), 2105, pp. 4552–4560.

[35] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, From contours to regions: An empirical evaluation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 2294–2301.

[36] L. Liu, L. Shao, P. Rockett, Boosted key-frame selection and correlated pyramidal motion-feature representation for human action recognition, Pattern Recognition 46 (7) (2013) 1810–1818.

[37] D. H. Hubel, T. N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, The Journal of Physiology 160 (1) (1962) 106–154.

[38] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biological Cybernetics 36 (4) (1980) 193–202.

[39] J. Wu, C. Leng, Y. Wang, Q. Hu, J. Cheng, Quantized convolutional neural networks for mobile devices, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4820–4828.

[40] E. P. Ijjina, K. M. Chalavadi, Human action recognition using genetic algorithms and convolutional neural networks, Pattern Recognition 59 (2016) 199–212.

[41] Y. Wang, C. Xu, S. You, D. Tao, C. Xu, Cnnpack: packing convolutional neural networks in the frequency domain, in: Advances in Neural Information Processing Systems, 2016, pp. 253–261.

[42] G. Gkioxari, J. Malik, Finding action tubes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 759–768.

[43] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[44] T. Y. Lin, A. Roychowdhury, S. Maji, Bilinear cnn models for fine-grained visual recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1449–1457.

[45] S. Zuffi, O. Freifeld, M. J. Black, From pictorial structures to deformable structures, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3546–3553.

[46] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, J. Davis, Scape: shape completion and animation of people, in: ACM Transactions on Graphics (TOG), Vol. 24, 2005, pp. 408–416.

[47] S. Zuffi, J. Romero, C. Schmid, M. J. Black, Estimating human pose with flowing puppets, in: IEEE International Conference on Computer Vision (ICCV), IEEE, 2013, pp. 3312–3319.

[48] T. Brox, A. Bruhn, N. Papenberg, J. Weickert, High accuracy optical flow estimation based on a theory for warping, in: European Conference on Computer Vision (ECCV), 2004, pp. 25–36.

[49] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, arXiv preprint arXiv:1405.3531.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 248–255.

[51] K. Soomro, A. R. Zamir, M. Shah, Ucf101: A dataset of 101 human actions classes from videos in the wild, arXiv preprint arXiv:1212.0402.

[52] Z. Xu, Y. Yang, A. G. Hauptmann, A discriminative cnn video representation for event detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1798–1807.

[53] J. Sánchez, F. Perronnin, T. Mensink, J. Verbeek, Image classification with the fisher vector: Theory and practice, International Journal of Computer Vision 105 (3) (2013) 222–245.

[54] H. Jégou, M. Douze, C. Schmid, P. Pérez, Aggregating local descriptors into a compact image representation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3304–3311.

[55] A. Vedaldi, B. Fulkerson, Vlfeat: An open and portable library of computer vision algorithms, in: Proceedings of the 18th ACM international conference on Multimedia, 2010, pp. 1469–1472.

[56] C.-C. Chang, C.-J. Lin, Libsvm: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 27.

[57] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, M. J. Black, Towards understanding action recognition, in: IEEE International Conference on Computer Vision (ICCV), 2013, pp. 3192–3199.

[58] B. Ni, V. R. Paramathayalan, P. Moulin, Multiple granularity analysis for fine-grained action detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 756–763.