Efficient Human Action Detection using a Transferable Distance Function

Weilong Yang, Yang Wang, and Greg Mori

School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
wya16@sfu.ca, ywang12@cs.sfu.ca, mori@cs.sfu.ca

Abstract. In this paper, we address the problem of efficient human action detection with only one template. We choose the standard sliding-window approach to scan the template video against test videos, and the template video is represented by patch-based motion features. Using generic knowledge learnt 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. Our method is evaluated on a human action dataset with cluttered background, and a ballet video with complex human actions. The experimental results show that our 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.

1 Introduction

This paper considers 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 our test videos. A closely related problem is action recognition, whose primary goal is to classify a video sequence into one of several pre-defined categories. The goal of action detection is distinct from that of action recognition – we would like to localize the specific position (in both time and space) of the target action in a video, rather than getting a single class label. 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.*

There is a large literature on action recognition and detection. Moeslund *et al.* [1] provide a survey of the literature on action recognition. We only give a brief review of the closely related work here. Niebles *et al.* [2] run an interest point detector over video sequences, then apply latent topic models to categorize and localize human actions. Ke *et al.* [3] apply the AdaBoost learning framework to the task of human action detection, and volumetric features are used for efficient video analysis. Laptev and Pérez [4] use a similar boosted space-time window classifier to localize human actions in movies. All these learning based

approaches heavily rely on a large number of training examples. 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 context of example-based video retrieval, we typically have only *one* short video clip submitted by the user.

One example of action detection with only one template is the work of Shechtman and Irani [5]. They compute the distance between two videos by exhaustively comparing patches centered around every space-time point. We use a similar patch based matching method, but we weight patches by their saliency, leading to a more efficient algorithm. In [6], Ke *et al.* propose a template matching scheme combined with a part based pictorial structure model to detect actions in crowded scenes with only one template. The limitation of this work is that one has to manually segment the parts (in space/time volumes), which can be time-consuming.

In our previous work [7], a patch based matching scheme is used for action recognition with a single clip as the template. We also propose a *transferable distance function* in [7] to weight those patches by their saliency. The transferable distance function is learnt from previously training sets, and can be applied to videos of new actions without further learning. The work presented here is based on [7]. However, in this paper, 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 [8, 7]. The main contributions of this paper are two-fold, in addressing the efficiency issues. First, we propose a variant of the motion feature in Efros *et al.* [8] 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 of [7], and significantly boosts the efficiency of our approach.

2 Human Action Detection

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, that is to slide the template action video clip T over all locations on the test video V. 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 proposed in [7]. However, we represent the motion feature using a histogram of four-channel motion flow, which enhances the efficiency of action detection.

2.1 Motion Feature

Our motion feature is a variant of the descriptor proposed by Efros *et al.* [8] 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 [8]. 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 [4]. The similarity between two feature vectors can be computed using normalized correlation or Euclidean distance. Moreover, to efficiently compute feature vectors, the *integral image* representation [9] is used for each histogram bin.

2.2 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.* [6] 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.

Following the work of [7], 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:

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

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.

3 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 [9] 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 [7]. We first define 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. In this section, we will review the learning of the transferable distance function, then introduce the construction of the cascade structure.

3.1 Transferable Distance Function

This idea of knowledge transfer has been exploited in the context of object recognition and identification [10, 11]. In particular, Ferencz *et al.* [10] propose to predict a patch's saliency for object identification by its visual feature called a *hyper-feature*. In human action recognition, we conjecture that there also exists a certain generic relationship between the saliency and the appearance of a patch [7]. For example, "stretched-arm-like" and "stretched-leg-like" patches are more likely to be salient than other patches. This generic relationship is "transferable", and we can employ this knowledge for patch weighting of unknown actions. In [7], we proposed the learning of a transferable distance function, which can extract generic knowledge of patch weighting from previous training sets, *e.g.* benchmark action datasets. When it is applied to unknown actions, the algorithm will look for salient patches and assign them high weights, that are also the parameters of the distance function for matching based recognition.

Given a patch i, the weight assigned to this patch is w_i, and we represent the hyper-feature of this patch as a $|V|$-dimensional vector $\mathbf{f}_i$ based on a codebook approach, where $|V|$ is the codebook size. The j-th element of $\mathbf{f}_i$ is set according to the distance between the feature vector of this patch and the j-th visual word. The feature vector of each patch consists of histogram of oriented gradient (HOG) [12] and patch positions. Please refer to [7] for more details. We assume that $\mathbf{f}_i$ and w_i have a linear relationship via the parameter $\mathbf{P}$:

$$w_i = \langle \mathbf{P} \cdot \mathbf{f}_i \rangle \tag{2}$$

Then we will have $\mathbf{w} = \mathbf{P}^T\mathbf{F}$, where each column of $\mathbf{F}$ denotes the hyper-feature vector of a patch, Each element of $\mathbf{w}$ denotes the weight of a patch. The objective is to learn $\mathbf{P}$ from the source training set. After the training, given any new action video, even if its action does not exist in the source training set, we can compute the weight of each patch of this video by Eqn. 2.

The learning of $\mathbf{P}$ follows the focal learning framework in [13]. The distance function obtained by $\mathbf{w} = \mathbf{P}^T\mathbf{F}$ will satisfy the constraints that the distance between dissimilar actions is larger than similar actions by the margin 1, that is $\langle \mathbf{w}_i \cdot (\mathbf{d}_{ij} - \mathbf{d}_{ik}) \rangle > 1$, $\langle \mathbf{P}^T\mathbf{F}_i \cdot (\mathbf{d}_{ij} - \mathbf{d}_{ik}) \rangle > 1$, 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. The weights are enforced to be non-negative, $\langle \mathbf{P} \cdot \mathbf{f}_m \rangle \geq 0$. For simplicity, we replace $\mathbf{d}_{ij} - \mathbf{d}_{ik}$ as $\mathbf{x}_{ijk}$. The max-margin optimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{P},\xi} \quad & \frac{1}{2}\|\mathbf{P}\|^2 + C\sum_{ijk}\xi_{ijk} \\ s.t.: \quad & \forall i,j,k: \quad \langle \mathbf{P}^T\mathbf{F}_i \cdot \mathbf{x}_{ijk} \rangle \geq 1 - \xi_{ijk} \\ & \forall m: \quad \langle \mathbf{P} \cdot \mathbf{f}_m \rangle \geq 0 \quad \forall i,j,k: \quad \xi_{ijk} \geq 0 \end{aligned} \tag{3}$$

where ξ_{ijk} is the slack variable and C is the trade-off parameter, similar to those in SVM. See [7] for more details about the solving of this optimization problem.

3.2 Construction of Cascade Structure

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 learnt distance function provides us a useful tool to obtain such a simple detector. After the learning on the source training set, we are able to compute the weights (*i.e.* saliency) of the patches on any given template action through Eqn. 2, and rank these patches by their saliency. 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:

$$D^k(T, L) = \sum_{i=1}^{M} \sum_{s \in E_i^k} w_{is} \min_{r \in R_s} d(t_{is}, q_{ir}) \tag{4}$$

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}.

In the cascade structure of [9], 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. 1. 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 4.2 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.

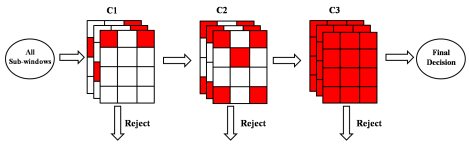

Fig. 1. An example of the cascade structure. The red patches are the effective patches on template frames. At the **C1** stage, the top-2 patches of each frame with high weights are used to match with the input sub-windows. At the **C2** stage, top-5 patches are used for matching. At the final stage, all patches are used.

4 Experiments

We evaluate our method on the cluttered human action dataset collected by Ke *et al.* [6], and a ballet video sequence. We first review the human action datasets, then present the experimental results.

4.1 Datasets

Weizmann Dataset [14]: 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 [7]. 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 [6]: 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.

4.2 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.* [6]. 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. 2. 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.

Fig. 2. Action detection examples on the cluttered action dataset. Representative frames of the template videos and the visualization of learnt 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.

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 step 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. 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. 2. As we can see, the high weights (red patches) are assigned to the *salient* parts, such as the stretched-arm, and bent-back.

After training, we can build the cascade structure based on the learnt 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 [6], 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.

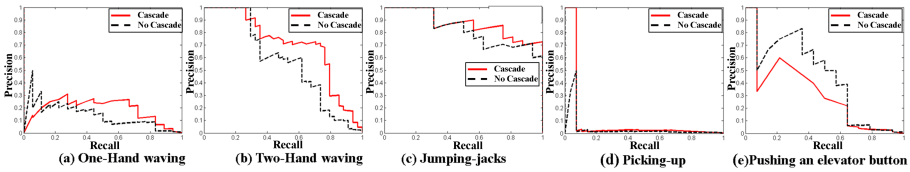

Fig. 3. Precision-Recall curves of the action detection on the cluttered action dataset.

Similar to [6], 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 Fig. 3. Since we choose a different way to scan the template over test videos, our results are not directly comparable with [6]. 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. Due to the large intra-class variation of actors performing the picking-up action, our method achieves very low detection rates on this action. 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. 2.

We give an example with more details in Fig. 4 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 actions cause very low matching distances but they are still much higher than the jumping-jacks action.

We also compare the results of using the distance function with and without thee cascade structure. As shown in Fig. 3, except for the action of pushing

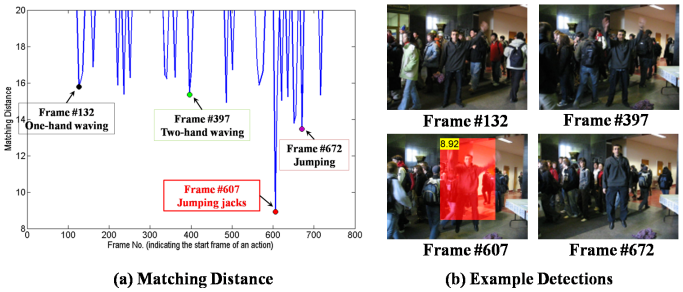

Fig. 4. (a) Projected matching distance of the detection of jumping-jacks. (b) Example detections. The true positives are highlighted in Frame #607, where the left corner is the matching distance. The rest frames are all true negatives.

an elevator button, our cascade structure achieves better accuracy. Moreover, the cascade structure is much more efficient. The methods are implemented in Matlab/MEX. With a 2.40GHz Intel processor, to scan a template video with 25 frames over a test video with 800 frames, the cascade structure only takes 30.3 seconds, but it takes 348.2 seconds without using the cascade. There is an order of magnitude improvement in efficiency by using the cascade structure.

4.3 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.* [6]).

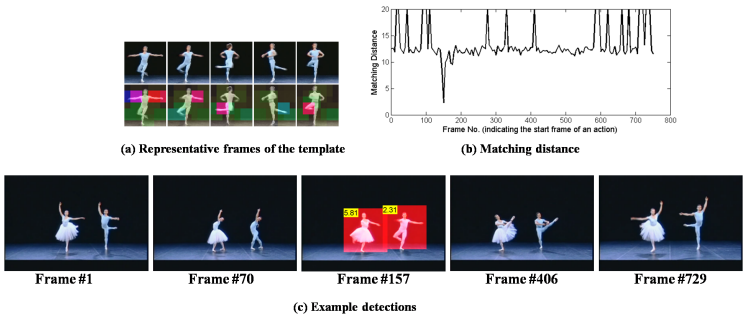

Fig. 5. (a) Representative frames of the template videos, and the visualization of learnt weights. (b) Projected matching distance. (c) Example detections. The true positives are highlighted in Frame #157, and the rest frames are all true negatives.

The Weizmann dataset serves as the source training set. The learnt weights on the template video are visualized in Fig. 5(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 salient 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.

5 Conclusion

In this paper, 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 [8], which can be efficiently computed using integral images. Based on the learning of a transferable distance function [7], 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.

References

1. Moeslund, T., Hilton, A., Kruger, V.: A survey of advances in vision-based human motion capture and analysis. CVIU **103**(2-3) (November 2006) 90–126
2. Niebles, J.C., Wang, H., Fei-Fei, L.: Unsupervised learning of human action categories using spatial-temporal words. IJCV **79**(3) (September 2008) 299–318
3. Ke, Y., Sukthankar, R., Hebert, M.: Efficient visual event detection using volumetric features. In: ICCV. Volume 1. (2005) 166–173
4. Laptev, I., Pérez, P.: Retrieving actions in movies. In: ICCV. (2007)
5. Shechtman, E., Irani, M.: Space-time behavior based correlation. In: CVPR. (2005)
6. Ke, Y., Sukthankar, R., Hebert, M.: Event detection in crowded videos. In: ICCV. (2007)
7. Yang, W., Wang, Y., Mori, G.: Human action recognition from a single clip per action. In: The 2nd International Workshop on Machine Learning for Vision-based Motion Analysis. (2009)
8. Efros, A.A., Berg, A.C., Mori, G., Malik, J.: Recognizing action at a distance. In: ICCV. (2003) 726–733
9. Viola, P., Jones, M.: Robust real-time face detection. In: IJCV. (2004)
10. Ferencz, A., Learned-Miller, E., Malik, J.: Learning to locate informative features for visual identification. IJCV (2006)
11. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. PAMI **28**(4) (April 2006) 594–611
12. Dalal, N., Triggs, B.: Histogram of oriented gradients for human detection. In: CVPR. (2005)
13. Frome, A., Singer, Y., Malik, J.: Image retrieval and classification using local distance functions. In: NIPS. Volume 19. MIT Press (2007)
14. Blank, M., Gorelick, L., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. In: ICCV. (2005)