Concurrent Action Detection with Structural Prediction

Ping Wei[1,2], Nanning Zheng[1], Yibiao Zhao[2], and Song-Chun Zhu[2]
[1]Xi'an Jiaotong University, China
[2]University of California, Los Angeles, USA
pingwei.pw@gmail.com,nnzheng@mail.xjtu.edu.cn
{yibiao.zhao,sczhu}@stat.ucla.edu

Abstract

Action recognition has often been posed as a classification problem, which assumes that a video sequence only have one action class label and different actions are independent. However, a single human body can perform multiple concurrent actions at the same time, and different actions interact with each other. This paper proposes a concurrent action detection model where the action detection is formulated as a structural prediction problem. In this model, an interval in a video sequence can be described by multiple action labels. An detected action interval is determined both by the unary local detector and the relations with other actions. We use a wavelet feature to represent the action sequence, and design a composite temporal logic descriptor to describe the action relations. The model parameters are trained by structural SVM learning. Given a long video sequence, a sequential decision window search algorithm is designed to detect the actions. Experiments on our new collected concurrent action dataset demonstrate the strength of our method.

1. Introduction

In the vision literature, action recognition is usually posed as a classification problem, i.e, a classifier assigns one action label to a video sequence [18]. However, action recognition is more than a classification problem.

First, a single human body can perform more than one actions at the same time. As Figure 1 shows, the person is *sitting* on the chair, *drinking* with the right hand, and *making a call* with the left hand, simultaneously. The three actions concurrently proceed forward in the time axis. In this case, the video sequence in the concurrent time interval can not be simply classified into one action class.

Second, multiple actions performed by one human body are semantically and temporally related to each other, as is shown in Figure 1. A person usually *sits* to *type on keyboard*, and rarely *stand* to *type on keyboard*. So the actions *sit* and *type on keyboard* semantically advocate each other while *stand* and *type on keyboard* are often exclusive. The action *turn on monitor* occurs usually *before* the action *type on keyboard*. Their locations and durations in the time axis are closely related. We believe that such information of action relations should play important roles in the action recognition and localization.

We define the *concurrent actions* as the multiple actions simultaneously performed by one human body. These actions can distribute in multiple intervals in a long video sequence, and they are semantically and temporally related to each other. By concurrent action detection, we mean to recognize all the actions and localize their time intervals in the long video sequence, as is shown in Figure 1.

In this paper, we propose a novel concurrent action detection model (COA). Our model formulates the detection of concurrent action as a structural prediction problem, similar to the multi-class object layout in still image [5]. In this formulation, the detected action instances are determined by both the unary local detectors and the relations with other actions. A multiple kernel learning method [2] is applied to mining the informative body parts for different action classes. With the informative parts mining, the human body is softly divided into the weighted parts which perform the concurrent actions. The parameters of the COA model are learned in the framework of structural SVM [17]. Given a video sequence, we propose an online sequential decision window search algorithm to detect the concurrent actions.

We collect a new concurrent action dataset for evaluation. Our dataset contains 3D human pose sequences captured by the Kinect camera [14]. It includes 12 action classes, which are listed in Figure 1, and totally 61 long video sequences. Each sequence contains many concurrent actions. The complex structures of the actions and the large noise of the human pose data make the dataset challenging. The experimental results on this dataset prove the strength of our method.

2. Related Work

Our work is related to four streams of researches in the literature.

(1) **Action recognition and detection** techniques have achieved remarkable progress in recent years [6, 8, 18, 20].

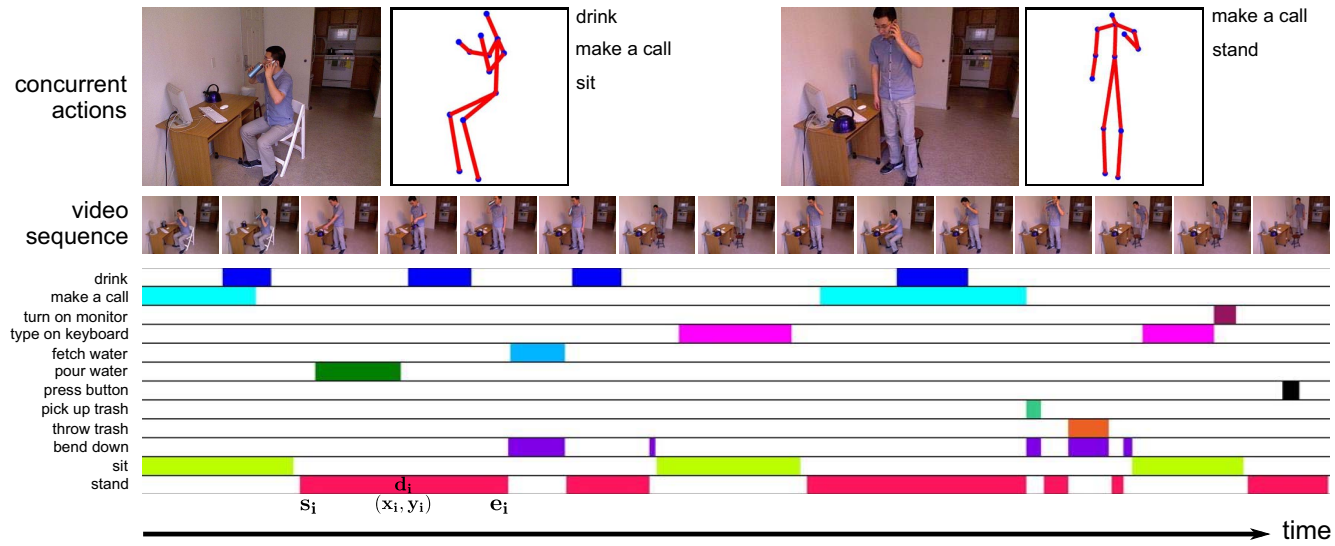

Figure 1. The illustration of the concurrent actions. Each horizontal row corresponds to an action class. The small colorful blocks correspond to the action intervals in the time axis.

Wang *et al*. [18] represented a 3D pose sequence by Fourier features and mined the actionlet ensemble with multiple kernel learning, which was then used to classify an new sequence. This method needs the video sequence to be presegmented, and predicts one action class for each segment. It is insufficient to interpret a video sequence with multiple concurrent and dependent actions. Hoai and Torre [6] trained an elaborate model to detect events in video before the events ended. However, it is focused on the early detection of an event and not applicable to detecting multiple concurrent actions.

(2) **Concurrent actions** exist in the literatures of other fields, like artificial intelligence [3, 12]. The work [12] represented the concurrent decision with a semi-Markov model where plans were learned from concurrent actions. These work are mainly for the robot planning, not for modeling the visual concurrent actions as in computer vision.

(3) **Temporal relations** are used in some literatures to facilitate the action modeling [1, 9, 10, 13, 16, 19]. The work [9, 13] decomposed an high level activity into partially ordered substructures which formed contexts of each other. And the work [13] suggested the actions could occur in parallel. However, they did not describe and learn the relations between different actions in a unified framework. Allen [1] introduced classical temporal logics to describe the relations between actions, which were further applied to representing the action structures and action detection in [10]. These temporal logics are qualitative descriptions, like *before, meet*, which are insufficient to describe complex relations with different degrees of overlapping intervals.

(4) **Structural prediction** has been used for object detection in still images. Desai *et al*. [5] modeled the multi-class object layout (MCOL) as a structural prediction problem. They trained the model with the structural SVM learning (SSVM) [17]. Our model is inspired by the MCOL and SSVM. But we modify and extend it to fit in with the motion data. Actually, the problem of action detection in motion data is more complex than the problem of object detection in still image because the motion data always has more data scales and more complex structures. Our COA model introduces new formulations to overcome these challenges.

3. Concurrent Action Model

Suppose there are M overlapping action intervals in a video sequence. These intervals are obtained by sliding the local action detectors of all the 12 action classes along the time axis in the video sequence, similar to the object detection in image with sliding windows. The ith interval is defined as $d_i = [s_i, e_i]$, where s_i and e_i are respectively the starting and ending time, as Figure 1 shows. x_i is the feature of the video clip in the interval d_i. $y_i \in \mathcal{Y}$ is the action class label of the interval d_i, where $\mathcal{Y}$ is the set of all action labels. The entire video sequence is encoded as the M action intervals, $X = \{x_i | i = 1, ..., M\}$. $Y = \{y_i | i = 1, ..., M\}$ is their label set. The score of interpreting the video sequence X with labels Y is defined as

$$S(X, Y) = \sum_i \omega_{y_i}^T \rho_{y_i}(x_i) + \sum_{(i,j)\in\mathcal{N}} \omega_{y_i,y_j}^T r_{ij} \quad (1)$$

where $i = 1, .., M, j = 1, ..., M$. $\rho_{y_i}(x_i)$ is the local detection model of the action y_i. It is a 2-dimension vector which encapsulates the local detection score and a constant 1 to adjust the bias. ρ_{y_i} is related to the action class y_i, which suggests that different actions correspond to different parts of the body. ω_{y_i} is the parameter of the action y_i.

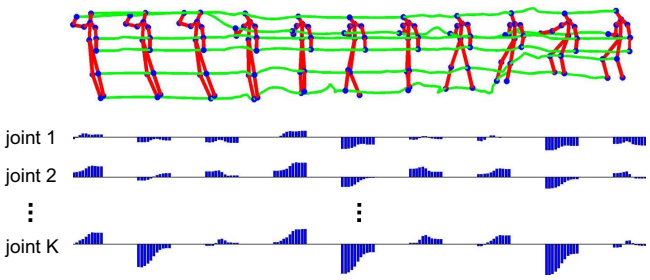

Figure 2. The wavelet feature of human action.

r_{ij} is the relation feature vector between the interval d_i and the interval d_j. ω_{y_i,y_j} is the relation parameter, which encodes the location and semantic relations between action classes y_i and y_j. $(i,j) \in \mathcal{N}$ means the interval d_i and d_j are neighbors. If the distance in the temporal axis between d_i and d_j is smaller than a threshold, then d_i and d_j are neighbors of each other. The introduction of the neighborhood system $\mathcal{N}$ indicates that an action in a sequence is only related to the actions which are close to it. This is because a video sequence can be very long. With the increase of the distance between two intervals, their dependent relations decrease.

The Eq.(1) is similar to the multi-class object layout model (MCOL) [5] in still image. However, our Eq.(1) introduces the neighborhood system into the structural prediction and accommodates the motion sequence data. Actually, our COA model is an extension of the MCOL model. If the size of the neighborhood is infinite, the Eq.(1) becomes the MCOL like in still image. If the size of the neighborhood is infinitesimal, the Eq.(1) shrinks to a local classifier model. The introduction of the neighborhood also raises the efficiency of inference. We will elaborate it later.

3.1. Wavelet Feature and Local Detection $\rho_{\mathbf{y_i}}(\mathbf{x_i})$

In our work, the input human action data is the sequence of 3D human poses which are estimated by the Kinect [14]. Each pose contains K 3D joint points of human body. A human action sequence forms K trajectories, as is shown in Figure 2. All the human poses are normalized by aligning the torsos and the shoulders. The estimated pose data is extremely noisy, which makes it very hard to characterize the action. It should be noted that though we use the 3D pose sequence as input in this work, our COA model is applicable in other sequence of human actions, like RGB video.

Wavelet was previously applied to representing the human motion feature [4, 11]. Inspired by them, we use the wavelet to describe the trajectories of the difference vectors between the 3D joints. These difference vectors present strong discriminative ability for action recognition [18]. Our objective is to extract robust and discriminative features for the sequence clip in the interval $[s,e]$. At time t, the relative location differences between the kth joint and all other joints are concatenated into a vector h_k^t. $\mathbf{h_k} = \{h_k^t | t = s, ..., e\}$ is the feature sequence of the kth joint in the interval $[s,e]$. $\mathbf{h_k}$ is a temporal signal in the interval $[s,e]$. It is interpolated into 128 frames. We apply the symlet wavelet transform to the interpolated $\mathbf{h_k}$, and keep the first V wavelet coefficients as the action feature of the kth joint, denoted as H_k. Then the sequence feature x of all the joints on the human body is $x = (H_1, ..., H_K)$.

With the wavelet feature x, the local action detection model is $\rho_{y_i}(x_i) = (f_{y_i}, 1)$, where f_{y_i} is an action detector:

$$f_{y_i} = \beta_{y_i}^T x + b_{y_i} \tag{2}$$

The wavelet transform has the attribute of time-frequency localization. It can extract the action's temporal structure. Also, the wavelet transform is multiscale. It can describe the action at different scales. Furthermore, by keeping the first V wavelet coefficients, we can eliminate the noise in the original pose data, which makes the action description more robust.

3.2. Composite Temporal Logic Descriptor for $\mathbf{r_{ij}}$

r_{ij} represents the temporal location of interval d_j relative to the interval d_i. In the famous work [1], Allen proposed 13 classical temporal relations between two intervals - *before, equal, meet, overlap, during, start, finish* and their inverses. These relations are qualitative descriptions, which cannot quantitatively describe the degree of temporal relations. For example, the action *press button* and *turn on monitor* both occur *before* the action *type on keyboard*. How do we measure and distinguish these two *before* relations?

We design a novel quantitative descriptor - composite temporal logic descriptor - to encode r_{ij}, as Figure 3 shows. It is decomposed into three components $r_{ij} = (r_{ij}^S, r_{ij}^C, r_{ij}^E)$:

1) r_{ij}^S, the location of d_j relative to the start point of d_i;

2) r_{ij}^C, the location of d_j relative to the center point of d_i;

3) r_{ij}^E, the location of d_j relative to the end point of d_i.

The first component r_{ij}^S encodes start relations between two actions. For example, human usually *bends down* to *pick up trash*. The action *bend down* and *pick up trash* always start simultaneously. So the action *pick up trash* is closely related to the start of *bend down*. r_{ij}^C encodes the entire relative location of two intervals.

The third component r_{ij}^E encodes the sequential relation of two intervals. For example, the action *throw trash* always occurs after the action *pick up trash* ends. So the action *throw trash* is closely related to the end of *pick up trash*.

We define a histogram with 8 uniform bins to describe the location of an interval relative to a time point. As Figure 3 shows, the 8 bins define 8 relations relative to the zero point O in the center of the histogram, *before-far*, *before-3*, *before-2*, *before-1*, *after-1*, *after-2*, *after-3*, *after-far*. The length of the histogram is set as 4 times the length of the

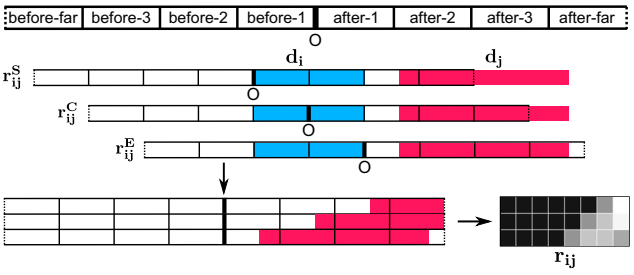

Figure 3. The composite temporal logic descriptor of d_j relative to d_i. The blue bar is the interval d_i. The red bar is the interval d_j.

interval d_i, which normalizes the histograms corresponding to different lengths of d_i.

To parameterize r^S_{ij}, we align the zero point O of the histogram to the start point of the interval d_i, as Figure 3 shows. We compute the duration of interval d_j falling in each bin of the histogram. The values of the bins *before-far* and *after-far* are the durations of interval d_j outside the *before-3* and *after-3*, respectively. These bin values are divided by the length of interval d_j to form the normalized descriptor r^S_{ij}. r^C_{ij} and r^E_{ij} are computed in a similar way but by aligning the zero point O to the center and the end of d_i, respectively.

Our descriptor decomposes the temporal relation into three components, which makes it able to describe subtle and complex temporal relations quantitatively. Because it quantizes the duration of action interval, it also characterizes the action's duration information.

4. Learning

4.1. Mining Informative Parts with MKL

This subsection elaborates on how we learn the local action detector $f_{y_i} = \beta^T_{y_i} x + b_{y_i}$ by mining the informative body parts for different actions. An action is usually related to some specific parts of human body. For example, the action *drink* is mainly performed by the hand and arms. The movements of other body parts, like legs and feet, are less important to this action. So for a specific action, the 'weight' of each body part is different. We use a multiple kernel learning (MKL) [2] method to automatically mine the informative parts for each action class. For clarity, we simplify y_i, β_{y_i}, and b_{y_i} as y, β, and b, respectively.

We introduce a weight vector $\alpha = (\alpha_1, ..., \alpha_K)$ for each action class y, where K is the number of human body joints, and $\alpha_k \geq 0$ corresponds to the kth joint. Each wavelet action feature x is decomposed into K blocks $x = (H_1, ..., H_K)$. The block H_k corresponds to the feature of the kth joint. The parameter β is correspondingly decomposed into the same format blocks as x, $\beta = (\beta_1, ..., \beta_K)$. Such decomposition makes it possible to differentiate the effects of different joints on the action y.

Suppose $\{(x_l, z_l)|l = 1, ..., L\}$ are L training samples for the action y, where z_l is the label of x_l. $z_l = 1$ if x_l is the positive sample of y, otherwise $z_l = -1$. Our goal is to learn the parameters (α, β, b) of the action y. This problem is formulated as a l_1-norm multiple kernel learning [2]:

$$\begin{aligned} \min \quad & \frac{1}{2}(\sum\nolimits_{k=1}^{K} \alpha_k ||\beta_k||_2)^2 + C \sum\nolimits_{l=1}^{L} \zeta_l \\ \text{w.s.t.} \quad & \alpha_k \geq 0, \zeta_l \geq 0, \beta, b \\ \text{s.t.} \quad & z_l(\beta^T x_l + b) \geq 1 - \zeta_l, \forall l \in \{1, ..., L\} \end{aligned} \tag{3}$$

This problem can be solved efficiently by the semi-infinite linear program [15].

4.2. Learning with Max-Margin Optimization

Given N action sequences $\{X_n|n = 1, ..., N\}$ and their manually annotated structural labels $\{Y_n|n = 1, ..., N\}$, the goal is to learn the parameter ω_{y_i} and ω_{y_i,y_j} in Eq.(1). Our learning formulation is based on the max-margin structural learning [5, 17]. We modify it to accommodate the sequential neighborhood-dependent data.

We rewrite the Eq. (1) as a compact form:

$$S(X, Y) = \omega^T \Phi(X, Y) \tag{4}$$

where

$$\omega = \begin{bmatrix} \omega_u \\ \omega_b \end{bmatrix}, \Phi(X, Y) = \begin{bmatrix} \sum_i \varphi(\rho_{y_i}(x_i), y_i) \\ \sum_{(i,j)\in\mathcal{N}} \psi(r_{ij}, y_i, y_j) \end{bmatrix} \tag{5}$$

ω_u and $\varphi(\cdot)$ are unary parameter and feature mapping vectors. ω_b and $\psi(\cdot)$ are binary parameter and relation mapping vectors. $\varphi(\cdot)$ is a $N_u A$ dimension vector which encapsulates A blocks, where A is the number of all action classes, and N_u is the dimension of feature $\rho_{y_i}(x_i)$. Each N_u-dimension block of ω_u corresponds to an action class. The elements of $\varphi(\rho_{y_i}(x_i), y_i)$ are all zeros except the block corresponding to the action class y_i, where it is $\rho_{y_i}(x_i)$. $\psi(\cdot)$ is a $N_b A^2$ dimension vector which encapsulates A^2 blocks, where N_b is the dimension of feature r_{ij}. Each N_b-dimension block corresponds to a pair of action classes. The elements of $\psi(r_{ij}, y_i, y_j)$ are all zeros except the block corresponding to the action class pair (y_i, y_j), where it is r_{ij}.

We formulate the parameter learning as a max-margin optimization [5, 17]:

$$\begin{aligned} \min_{\omega, \xi_n \geq 0} \quad & ||\omega||^2 + C \sum\nolimits_{n=1}^{N} \xi_n \\ \text{s.t.} \quad & \forall n = 1, ..., N, \forall \hat{Y}_n, \\ & \omega^T \Delta(X_n, Y_n, \hat{Y}_n) \geq \delta(Y_n, \hat{Y}_n) - \xi_n \end{aligned} \tag{6}$$

where $\hat{Y}_n$ is the false structural label of the sequence X_n. $\delta(Y_n, \hat{Y}_n)$ is a 0-1 loss function $\delta(Y, \hat{Y}) = \sum_{i=1}^{|Y|} \mathbf{1}(y_i \neq \hat{y}_i)$, where $|Y|$ is the dimension of Y.

$\Delta(X_n, Y_n, \hat{Y}_n) = \Phi(X_n, Y_n) - \Phi(X_n, \hat{Y}_n)$ is the difference between compact features with the true label and the false label. The inequation in model (6) means that in all training sequences, the score of the true label should be larger than all other false labels by a soft margin.

The problem (6) can be solved by a cutting-plane algorithm [7]. Our model introduces the neighborhood to the compact feature $\Phi(\cdot)$ in Eq.(5). It reduces the search space when solving the optimization problem.

5. Inference

Given a long temporal sequence X containing multiple actions, our goal is to localize all the action intervals and label them with the action classes. It is formulated as:

$$Y^* = \text{argmax}\, S(X, Y) \tag{7}$$

The work [5] adopted a greedy search algorithm to solve the NP-hard problem (7). It demonstrated that though the greedy search algorithm produced suboptimal solutions, it was effective for object layout in the image. The detection of multiple concurrent actions in temporal sequence is more complex than the object layout in the still image. The image plane is limited, which makes it possible to search the solutions in a tolerable period. However, a temporal sequence can be very long and contain large number of actions, which makes the normal greedy search inapplicable. We propose an sequential decision window search algorithm to solve this problem (7), which extends the normal greedy search algorithm [5] to the sequential data with large durations.

We introduce a temporal window W. It slides by a smaller step than the size of itself, from the start of the sequence to the end, which generates a series of overlapping windows, $\{W_t | t = 1, 2, ...\}$. We call them decision windows. In each decision window, we carry out the greedy search algorithm based on the optimized results in the previous decision windows. With the decision window sliding forward, the entire sequence is structurally labeled.

We first run the local detectors (Eq.(2)) of all the 12 action classes on the temporal sequence in a sliding-window manner. For each action class, we run multiple detectors with multi-scales. Such local detection process produces a large amount of action intervals, which are pruned by a non-maxima suppression step to generate M' hypothesized action intervals $D = \{d_i | i = 1, ..., M'\}$.

Suppose $D_s \subseteq D$, and X_{D_s} and Y_{D_s} are respectively the feature set and the corresponding action label set of the action intervals in D_s. We define the score of the subset D_s as $S(D_s) = S(X_{D_s}, Y_{D_s})$, and $S(D_s) = 0$ when D_s is empty. We want to select a subset D_s from D that $S(D_s)$ achieves the maximum value in all subsets of D. We define $D_u = D - D_s$ is the set of unselected intervals, and $D_w = D_u \wedge W_t$ is the set of unselected intervals located in the decision window W_t. We define a score change after a new interval d is added to D_s: $\Delta(d) = S(D_s \cup \{d\}) - S(D_s)$. With these notations, our sequential decision window search algorithm is summarized in Algorithm 1.

Algorithm 1 Sequential Decision Window Search

Initialization:

$t = 1$, $D_s = \{\}$, $D_u = D$, $D_w = \{\}$;

Iteration:

1: **Decision window forward**

$D_w = D_u \wedge W_t$;

2: **Greedy search in decision window**

(i) $d^* = \text{argmax}_{d \in D_w} \Delta(d)$;

(ii) **if** $\Delta(d^*) < 0$, break and go to step 3;

else, $D_s = D_s \cup \{d^*\}$

$D_u = D_u - \{d^*\}$

$D_w = D_w - \{d^*\}$

(iii) **if** D_w is empty, break and go to step 3;

else, go to step (i);

3: **if** W_t arrives at the sequence end, **stop** and **output** D_s;

else, t = t + 1, go to step 1.

Our sequential decision window search is the general case of the normal greedy search algorithm [5]. If the size of the decision window is set to be the duration of the entire sequence, it becomes the global greedy search.

In general cases, our search algorithm is suboptimal compared to the normal greedy search. But it is reasonable in the human action sequence data because an action is usually only related to other actions which are close to it. Our experimental results also prove its effectiveness and reasonability.

In our algorithm, the decision window slides from the sequence beginning to the end. This makes it possible to detect the actions online. This advantage is especially useful in the practical applications, like video surveillance, robot navigation, and human-computer interactions.

6. Experiment

6.1. Dataset

To evaluate our method, we collect a new concurrent action dataset with annotation. The dataset is captured using the Kinect camera [14], which estimates the 3D human skeleton joints at each frame. Several volunteers are asked to perform actions freely in the daily-life indoor scenes, like office and living room. The action orders, poses, durations, and numbers are all decided according to their personal habits. Totally, we collected 61 long video sequences. Each sequence contains many actions which are concurrent in the time axis and interact with others. The dataset includes 12 action classes: *drink, make a call, turn on moni-*

Action	SVM -SKL	SVM -WAV	ALE [18]	MIP	Our COA
drink	0.77	0.70	0.91	0.92	**0.96**
make a call	0.75	0.86	0.85	0.93	**0.97**
turn on monitor	0.40	0.34	**0.55**	0.42	0.43
type on keyboard	0.82	0.91	0.92	0.91	**0.93**
fetch water	0.40	0.23	0.58	0.59	**0.60**
pour water	0.66	0.70	**0.71**	0.58	**0.71**
press button	0.17	0.20	**0.66**	0.22	0.33
pick up trash	0.39	0.35	0.39	0.40	**0.55**
throw trash	0.11	0.33	0.21	0.29	**0.59**
bend down	0.32	0.65	0.47	0.58	**0.67**
sit	0.98	**0.99**	**0.99**	0.98	0.98
stand	0.86	0.90	0.95	0.96	**0.97**

Table 1. The average precision comparison on each action class.

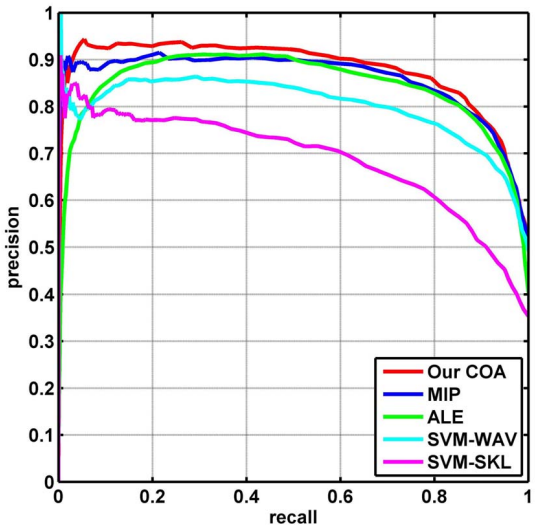

Figure 4. The precision-recall curves on the entire test dataset.

SVM-SKL	SVM-WAV	ALE [18]	MIP	Our COA
0.69	0.80	0.84	0.86	**0.88**

Table 2. The overall average precision comparison.

tor, type on keyboard, fetch water, pour water, press button, pick up trash, throw trash, bend down, sit, and *stand*.

Our dataset is new in two aspects: i) each sequence contains multiple concurrent actions; ii) these actions semantically and temporally interacts with each other. Our dataset is challenging. Firstly, the human skeleton estimated by the Kinect is very noisy. Secondly, the duration of each sequence is very long. Thirdly, the instances of each action class have large variances. For example, some instances of the action *sit* last for less than thirty frames, but some may last for more than one thousand frames. Finally, some different actions are very similar, like *drink* and *make a call*, *pick up trash* and *throw trash*.

6.2. Concurrent Action Detection

Evaluation criterion. A detected action interval is taken as correct if the overlapping length of the detected interval and the ground truth interval is larger 60% than their union length or the detected interval is totally covered by the ground truth interval. The second condition is special in action detection because part of an action is still described with the same action label by human. We measure the performance with the average precision (AP) of each class, and the overall AP on the entire testing data.

Baseline. We compare our model (COA) with four baselines. (1) SVM-SKL. This method uses the original aligned skeleton sequence as the action feature, and a SVM trained detector to detect the action with sliding windows. (2) SVM-WAV. This method is similar to the SVM-SKL except for that its action feature is our proposed wavelet feature. (3) ALE. Actionlet ensemble [18] is the state-of-art method in multiple action recognition with the 3D human pose data. It achieves the highest performance on many dataset compared to the previous best results. We train it as a binary classifier and test it on our dataset under the sliding window detection framework. (4) MIP. This is our local detector (Eq. 2) with mining informative parts. It is part of our COA model without using the temporal relations between actions. The originally detected intervals of the four methods are processed with the non-maxima suppression to output the final results.

The AP of each class. Table 1 shows the average precision of each action class. In most action classes, our method outperforms the other methods, which proves its effectiveness and advantage. Some actions are hard to be detected just by the independent local detector. The temporal relation between them and other action classes can facilitate the detection. For example, the action *throw trash* is usually inconspicuous and hard to be detected. With the context of *pick up trash* which usually occurs closely before *throw trash*, the AP of *throw trash* is significantly boosted. Reciprocally, the precision of *pick up trash* is also jointly improved by the context of *throw trash*.

The overall AP. We also compute the overall average precision, i.e., the results of all the testing sequences and all the action classes are put together to compute the AP. It measures the overall performance of each algorithm. Figure 4 shows the precision-recall curves of all the methods. Table 2 presents the overall average precision. Our model presents better performance than the other methods.

The SVM-WAV and the SVM-SKL are different in the action sequence feature. The better performance of the SVM-WAV than the SVM-SKL proves that our wavelet feature is more descriptive than the 3D human pose feature. The MIP and the SVM-WAV use the same wavelet feature but different learning method. The better performance of

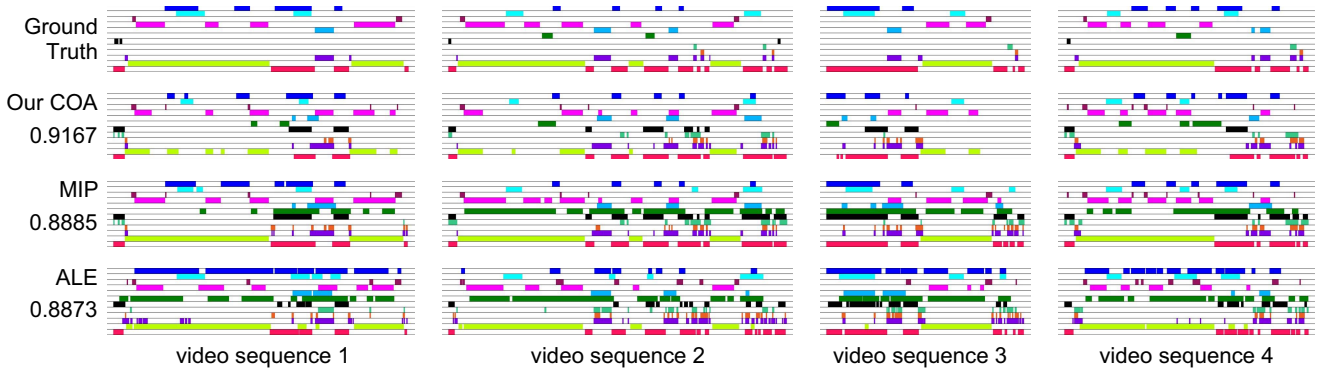

Figure 5. The concurrent action detection results in four sequences. Each horizontal row in an bar-image corresponds to an action class. The small colorful blocks are the action intervals. The numerical values are the average overlapping rates of each method's bar-images with the ground truth images. The rates show that the results of our COA model are closer to the ground truth than other methods.

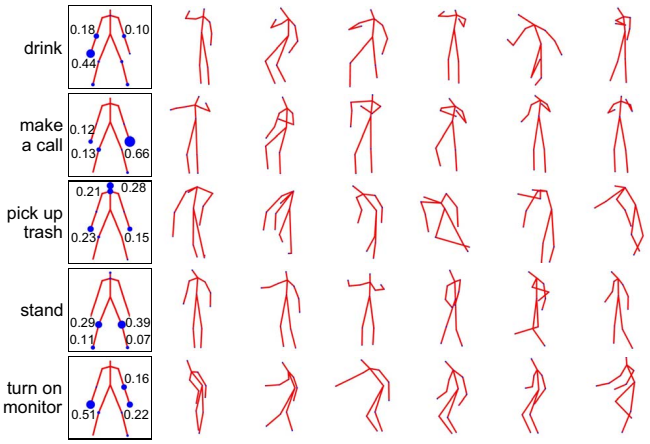

Figure 6. The informative body parts for some actions. The first column is the learned informative body parts. The areas of the joints correspond to the magnitude of the weight. Other poses are the instances of the action. For clarity, we just label the joints with larger weight. The joints on shoulder and torso are the reference for the pose alignment, and therefore are not attached the weights.

the MIP than the SVM-WAV proves the strength of our informative parts mining method. Our COA model achieves better performance than MIP, which demonstrates the effect of the temporal relations between actions.

The visualization of the detection. To intuitionally shows the strength of our model, we visualize some action detection results in Figure 5. We compare them with the results of the two best baselines, the ALE [18] and MIP. We also compute the average overlapping rate of each method's results with the ground truth. From the comparison, we can see that our COA model can remove many false positive detections with the action relations.

6.3. Informative Body Parts

The informative body parts are weighted human body parts for different action classes. We visualizes the learned weights of human body joints (the normalized weights of multiple kernels [15]) in Figure 6.

An action is usually related to some specific parts of human body. And other body parts are less relevant to this action. Our multiple kernel learning method can automatically learn these informative body parts. Figure 6 shows that though the data of action instances is noisy and has large variance, our algorithm can mine the reasonable body parts for different action classes.

6.4. Temporal Relation Templates between Actions

The composite temporal logic descriptor represents the co-occurrence and location relations between actions. We learn these temporal relation parameter ω_{y_i,y_j} from our manually labeled dataset. This parameter is like a template, which encodes the weight of temporal relations between actions. We visualize the learned parameter in Figure 7.

From this figure, we can see that our composite temporal logic descriptor and the learning method reasonably capture the co-occurrence and temporal location relations between actions. For example, the action *throw trash* usually occurs after the action *pick up trash*. So the weights of the bins encoding the *after-far* relations are larger than other bins. The action *type on keyboard* usually co-occurs with the action *sit*. So the weights of the middle bins are much larger than the weights of the *before* or *after* parts. The uniform blocks represents the independence or small dependence of two actions, like the relation between *fetch water* and *make a call*.

Another advantage of our descriptor is that it can characterize the duration relations of actions, which is important information of an action. This is displayed by that the descriptor of action y_j to y_i and the descriptor of y_i to y_j are unsymmetrical, as the relations between *turn on monitor* and *type on keyboard*. This is because our descriptor is related to the location of the start, center, and end of the reference action, not only dependent on one location point.

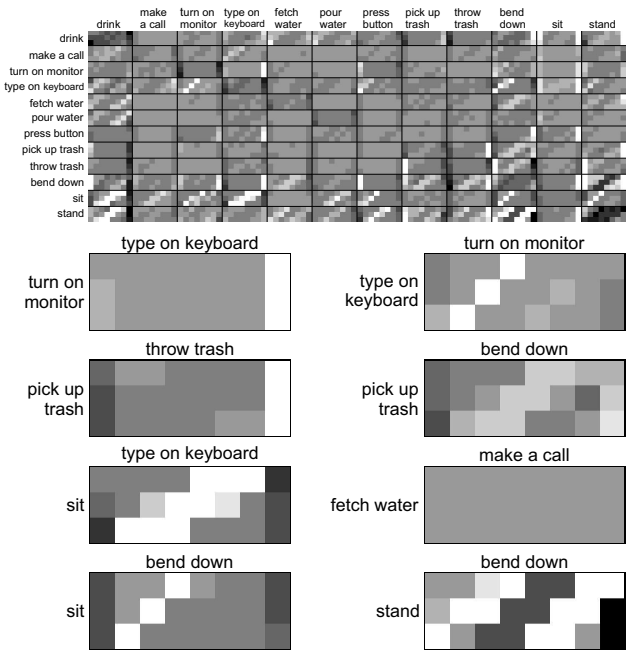

Figure 7. The learned temporal relation templates. The pairwise relation between two actions is shown as a 3×8 block. The three rows correspond to r_{ij}^S, r_{ij}^C, and r_{ij}^E, respectively. Each block describes the relation of the column action relative to the row action. The brighter colors correspond to the larger values of the weight.

7. Conclusion

In this paper, we present a new problem of concurrent action detection and proposes a structural prediction formulation for this problem. This formulation extends the action recognition from unary feature classification to multiple structural labeling. We describe the phenomenon of the concurrent actions by introducing the informative body parts, which are mined for each action class by multiple kernel learning. To accommodate the sequential nature and large duration of video sequence, we design a sequential decision window search algorithm, which can online detect actions in video sequence. We design two descriptors for representing the local action feature and temporal relations between actions, respectively. The experiment results on our new concurrent action dataset demonstrate the benefit of our model. The future work will focus on the multiple action detection in real surveillance video of large scenes.

Acknowledgement

The authors thank the support of grant: ONR MURI N00014-10-1-0933, DARPA MSEE project FA 8650-11-1-7149, and 973 Program 2012CB316402.

References

[1] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[2] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. ICML, 2004.

[3] C. Boutilier and R. I. Brafman. Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14(1):105–136, 2001.

[4] W. Chen and S.-F. Chang. Motion trajectory matching of video objects. In *SPIE Proceedings of Storage and Retrieval for Media Databases*, 2000.

[5] C. Desai, D. Ramanan, and C. C. Fowlkes. Discriminative models for multi-class object layout. *International Journal of Computer Vision*, 95(1):1–12, 2011.

[6] M. Hoai and F. De la Torre. Max-margin early event detectors. In *CVPR*, 2012.

[7] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[8] M. Müller and T. Röder. Motion templates for automatic classification and retrieval of motion capture data. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006.

[9] M. Pei, Y. Jia, and S.-C. Zhu. Parsing video events with goal inference and intent prediction. In *ICCV*, 2011.

[10] C. S. Pinhanez and A. F. Bobick. Human action detection using pnf propagation of temporal constraints. In *CVPR*, 1998.

[11] K. Quennesson, E. Ioup, and C. L. Isbell. Wavelet statistics for human motion classification. In *AAAI*, 2006.

[12] K. Rohanimanesh and S. Mahadevan. Learning to take concurrent actions. In *NIPS*, 2002.

[13] Y. Shi, Y. Huang, D. Minnen, A. F. Bobick, and I. A. Essa. Propagation networks for recognition of partially ordered sequential action. In *CVPR*, 2004.

[14] J. Shotton, A. W. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011.

[15] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[16] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012.

[17] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[18] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, 2012.

[19] P. Wei, Y. Zhao, N. Zheng, and S.-C. Zhu. Modeling 4d human-object interactions for event and object recognition. In *ICCV*, 2013.

[20] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, 2009.