Discriminative Orderlet Mining For Real-time Recognition of Human-Object Interaction

Gang Yu[1]*, Zicheng Liu[2], Junsong Yuan[1]

[1]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
gyu1@e.ntu.edu.sg, jsyuan@ntu.edu.sg

[2]Microsoft Research
Redmond, WA, USA
zliu@microsoft.com

Abstract. This paper presents a novel visual representation, called orderlets, for real-time human action recognition with depth sensors. An orderlet is a middle level feature that captures the ordinal pattern among a group of low level features. For skeletons, an orderlet captures specific spatial relationship among a group of joints. For a depth map, an orderlet characterizes a comparative relationship of the shape information among a group of subregions. The orderlet representation has two nice properties. First, it is insensitive to small noise since an orderlet only depends on the comparative relationship among individual features. Second, it is a frame-level representation thus suitable for real-time online action recognition. Experimental results demonstrate its superior performance on online action recognition and cross-environment action recognition.

1 Introduction

Human movement exhibits strong coordination patterns among the skeleton joints. Each type of action typically involves a subset of joints, and the spatial configurations of these joints are strong characteristics of the action. For example, when people talk over phone, the hand that holds the phone is usually close to the ear no matter whether the person is sitting, bending, standing, or walking. This particular spatial configuration between the hand and the ear is thus a characteristic of the talking-over-phone action. We believe that if we can model such action-dependent inter-joint coordination patterns, it will provide us with an effective tool for action recognition. This paper is one step along this direction.

We propose to use an ordinal representation, called orderlets, to encode the s-patial configuration of a group of skeleton joints. Generally speaking, an orderlet captures the ordinal information among a group of primitive feature values. For example, if we use the X-coordinate as the primitive feature, then the ordinal information represents which joint is the leftmost and which joint is the rightmost. If we use the pairwise Euclidean distances between the joints as the primitive features, then the ordinal information represents which joint pair is the closest

* The work was done when Gang Yu was an intern at Microsoft Research. This work is supported in part by Singapore MoE Tier-1 grant.

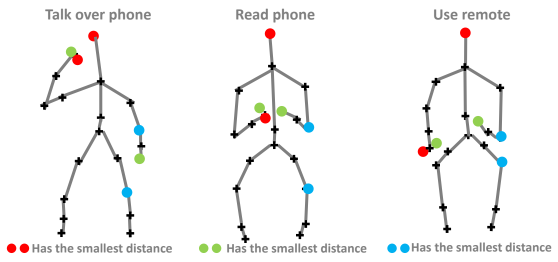
Talk over phone
Read phone
Use remote
Has the smallest distance
Has the smallest distance
Has the smallest distance

Fig. 1. An illustration of our orderlet representation.

and which joint pair is the farthest apart. Again consider the talking-over-phone action. Instead of requiring the distance between the ear and the hand to be a small value, we can require the hand-ear distance to be smaller than the distance between left hand and right hand. One example can be found in Fig. 1.

Such ordinal representation is also applicable to shape features to describe object shape information. Given a hypothesized object patch, we can divide the patch into multiple subregions. The comparative relationship between the shape features in the subregions can be captured by orderlets. In this way, both the skeleton information and the object shape information can be represented by orderlets and combined to recognize human-object interactions.

The orderlet representation has the following two properties. First, it is insensitive to small noise. An orderlet only depends on the comparative relationship among the primitive feature values. Such information is less sensitive to noise than the numerical values. Also, it is capable of handling missing or incorrect joints caused by occlusions or skeleton tracking module. Many orderlets only depend on a small group of skeleton joints. They can be correctly detected as long as the joints that they depend on are not missing. Second, the orderlet representation is well suited for real-time online action recognition from unsegmented streams since an orderlet is a frame-level representation. We have developed a real-time online action recognition system from a commodity depth sensor. The system does not require temporal segmentation, and it can handle natural transitions between two consecutive actions. Our contributions can be summarized as follows:

- We propose a novel middle level representation, called orderlets, for action recognition. It is robust to noises and missing joints, and is flexible to handle large intra-class variations.
- An orderlet mining algorithm is presented to effectively discover the discriminative orderlets from a large pool of candidates.
- We build a real time system that continuously recognizes human-object interactions using a RGB-D camera. To evaluate the performance, we collect a new dataset that contains both segmented video sequences for offline action recognition and unsegmented video sequences for online action recognition.

2 Related Work

In the past decade, there has been tremendous amount of work on human action recognition and detection from static images and 2D video sequences. It is impossible to list them all, and we only mention a few [25, 26, 28–30, 10, 11, 31, 32]. Recently, with the development of the commodity depth sensors like Kinect, there has been a lot of interests in human action recognition from depth data [4, 12, 24]. Like other visual recognition tasks, the performance of action recognition depends on the visual representation. In [13], spatial-temporal interest points are proposed to represent an action where histogram of gradients and histogram of optical flows are used to describe the extracted interest points [9]. Different from sparse interest points as in [13], a dense cuboid feature is presented in [14] for action recognition. Recently, dense trajectory [15, 16] based algorithm achieved promising results on the challenging datasets like HMDB and Hollywood2.

For depth data, [1, 19, 21, 27] utilize skeleton information as features for action recognition. In [4], it proposed the actionlet ensemble framework. Our work differs from [4] in two key aspects. First, an actionlet does not encode the spatial relationship between its joints. Second, the actionlet ensemble is a sequence-level representation that requires a segmented sequence for feature extraction and recognition. It is not suitable for online action recognition from unsegmented streams.

Instead of relying on skeleton information, many researchers have developed features based on depth maps. [5] proposed to randomly sample a large number of occupancy features in the 4D space of a depth sequence. Encouraged by the work of using 3D normal vectors for object recognition [7], [12] proposed the histogram of oriented 4D normal features. [17] extended the spacetime interest point features to the depth sequences, and developed a filtering technique to suppress the noises in the depth maps thus improving the quality of the interest point detection. Other interesting action representations based on depth streams include [20, 22].

Order representation has been widely used in image indexing and search. In [2], min-hash is employed for near-duplicate image search. Winner-take-all hash is presented in [3] for image search. A set of random patterns are generated and the element with the minimum value is encoded. However, there are two main differences between our work and the previous order representations. First, the previous order representations fix the group size of the primitive features, which is the number of items in an orderlet. This is not suitable for action recognition because different actions may involve different number of joints. In contrast, we allow arbitrary group size in the orderlets. Second, the order features are usually randomly generated in the previous work [3] while we use a data mining process to select the discriminative orderlets.

The orderlet representation is related to the attribute representation [8, 33] in that both are higher level representations compared to the basic low-level features. There are two main differences between the orderlets and attributes. First, attribute is a sequence-level representation while orderlet is a frame-level representation. Thus orderlets are more suitable for online action recognition.

Second, attributes can be designed manually as demonstrated by [8] where manually designed attributes are important for action recognition performance. In contrast, all the orderlets are discovered automatically by a data-driven process.

3 Orderlet

Suppose we have a training dataset with N_R videos: $\mathcal{R} = \{(\mathcal{V}_i, y_i), i = 1, 2, \cdots, N_R\}$, where $y_i \in \{0, 1\}$ refers to the label of the video. $\mathcal{V} = [I_1, I_2, \cdots, I_T]$ refers to a video sequence from a RGB-D camera, where $I_t, t = 1, 2, \cdots, T$, is a video frame.

3.1 Primitive Skeleton Feature

The pose of human skeleton provides strong cues to recognize human-object interaction, as illustrated in Fig. 1. Thanks to [1], skeleton feature now can be extracted from a RGB-D video sequence. For frame I_t, the skeleton information is denoted as $\mathbf{S}^t = \{\mathbf{s}_1^t, \mathbf{s}_2^t, \cdots, \mathbf{s}_{N_S}^t\}$, where $\mathbf{s}_i^t = (x_i^t, y_i^t, z_i^t)$ refers to the coordinate position of joint i on the t-th frame and $N_S = 20$ is the total number of joints.

We propose three types of primitive features extracted from human skeleton:

– Pairwise joint distance:
$$\lambda^{(1)} = ||\mathbf{s}_i^t - \mathbf{s}_j^t||. \tag{1}$$
– Spatial coordinate of a joint:
$$\lambda^{(2)} = x_i^t \ \ or \ \ y_i^t \ \ or \ \ z_i^t. \tag{2}$$
– Temporal variation of a joint location:
$$\lambda^{(3)} = ||\mathbf{s}_i^t - \mathbf{s}_i^{t-\Delta}||, \tag{3}$$
where Δ denotes a time duration.

Pairwise joint distance feature (Eq. 1) can be used to describe the distance between joint pair in one frame. One example can be found in Fig. 1 where the distance between each pair of dots can be considered as one primitive feature. Spatial coordinate feature (Eq. 2), on the other hand, is to describe the joint's coordinate position. For example, $\lambda^{(2)} = z_i$ can be used to indicate which joint is close to the camera. Finally, temporal variation of a joint location (Eq. 3) describes the motion of one joint given a time period Δ. In general, primitive features in Eq. 1, Eq. 2 and Eq. 3 are independent of time t.

We further denote $\Lambda^{(f)}$ as the complete primitive features of type f. Then, we have $|\Lambda^{(1)}| = N_S \times (N_S - 1)/2$, $|\Lambda^{(2)}| = N_S \times 3$, and $|\Lambda^{(3)}| = N_S \times N_\Delta$, where N_Δ refers to the number of time durations for Eq. 3.

3.2 Order Representation

There are large intra-class variations in the human actions. The same actions performed by the same person may differ a lot in terms of speed and style, not to mention the actions performed by different people in different environments. As a result, using the raw values of these primitive features is not robust.

Motivated by Fig. 1, instead of computing the raw values from the skeleton primitive features, a small set of joints with special ordinal configuration should be more meaningful for action recognition. As illustrated in Fig. 1, three joint pairs are selected with red, green and blue color, respectively. Although the raw values of the pairwise joint distance in Eq. 1 may be noisy for action classification, the ordinal configuration among the three joint pairs is actually stable for each specific human behavior. For example, for the talking-over-phone action, the red joint pair has the smallest distance among the three pairs while, for reading phone and using remote actions, the green and blue joint pairs should have the smallest distance, respectively.

Order representation has been successfully employed in image indexing and searching [2][3]. Inspired by the previous work, we propose a novel middle level representation, called orderlets. Formally, we define a size-n orderlet $\mathbf{p}$ as:

$$\mathbf{p} = (O_{\mathbf{p}}, k), \tag{4}$$

where $O_{\mathbf{p}} = [\lambda_{i_1}^{(f)}, \lambda_{i_2}^{(f)}, \cdots, \lambda_{i_n}^{(f)}]$ is a subset from $\Lambda^{(f)}$ and k is an index value for the element of $O_{\mathbf{p}}$ with the minimum value. f is denoted as the primitive feature category. The response of orderlet $\mathbf{p}$ on video frame I^t is defined as

$$\mathbf{v}_{\mathbf{p}}(I^t) = \begin{cases} 1 & \lambda_{i_k}^{(f)} \leq \lambda_{i_j}^{(f)} \text{ for all } \lambda_{i_j}^{(f)} \in O_{\mathbf{p}} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Given a video sequence $\mathcal{V}$ with T frames, the representation of the video $\mathcal{V}$ based on orderlet $\mathbf{p}$ is:

$$\mathbf{V}_{\mathbf{p}}(\mathcal{V}) = \sum_{t=1}^{T} \mathbf{v}_{\mathbf{p}}(I^t). \tag{6}$$

3.3 Object Feature

Only using the skeleton feature is not sufficient to address the human-object interaction recognition problem. For example, it is difficult to predict the action if the actor is lifting his hand near the head position. It could be an eating action if food is held in the hand, or a picking up phone call action if the phone is in his hand. Thus, it is critical to utilize the object information for the action classification.

Different from previous work, which either extracts the object feature from the neighborhood of each joint position [4] or samples random patterns from the RGB-D space [5], we propose to focus on the potential object positions. Given the skeleton information, it is easy to model the object position relative to skeleton joints for each action class. More specifically, during the training stage, the distance between hand (both left and right) and object center is utilized for each frame and a clustering is performed to obtain several frequent hand-object shifts. Also, frequent object sizes can be obtained by a clustering process on the objects from the training data. For each testing frame, these frequent shifts and scales can be used to generate a set of potential object positions.

For each potential object position, Local Occupancy Pattern (LOP) [4] is extracted to obtain the object shape information from the depth video. Specifically, for each potential local region, it is partitioned into a grid with $N_b = N_x \times N_y \times N_z$ non-overlapping cells. The number of cloud points is computed for each cell, denoted as γ, and a sigmoid function is employed to obtain the occupancy information $l = \frac{1}{1+exp(-\beta\gamma)}$, where β is a parameter. The concatenation of all the occupancy information from each cell is the LOP feature: $\mathbf{d} = [l_1, l_2, \cdots, l_{N_b}]$.

Similar to skeleton feature in Section 3.2, the extracted LOP feature is further encoded by the order representation. The LOP primitive feature can be defined as:

$$\lambda^{(4)} = ||\mathbf{d}(i) - \mathbf{d}(j)|| = ||l_i - l_j||, \quad 1 \leq i, j, \leq N_b, \quad i \neq j. \tag{7}$$

In total, we have $|\Lambda^{(4)}| = N_b \times (N_b - 1)/2$ such features. Based on the LOP primitive feature in Eq. 7, we define LOP orderlets as in Eq. 4 of Section 3.2. LOP orderlets can represent the comparative relationship among the primitive shape feature of a subset of the grids, which is an important complementary feature when skeleton is ambiguous or noisy. Based on the skeleton and object orderlet representation, the next section will discuss how to obtain discriminative orderlets for action recognition.

3.4 Orderlets Discovery

We present a feature mining approach to discover a pool of discriminative orderlets, whose response $V_{\mathbf{p}}$ in Eq. 6 is high for the positive videos but low for the negative ones.

Initially, we have different kinds of primitive features, either from skeleton feature or object feature. Let us take the pairwise joint distance feature (Eq. 1) as an example. Suppose we have N_S joints in the skeleton, then the total number of pairwise joint distance is $|\Lambda^{(1)}| = N_S \times (N_S - 1)/2$. We start from size-2 orderlet, which can be enumerated since the total number of candidates is $|\Lambda^{(1)}| \times (|\Lambda^{(1)}| - 1)/2$.

Given a size-2 orderlet $\mathbf{p}$ and a threshold $\theta_{\mathbf{p}}$, we can define a classification function $\mathcal{F}_{\mathbf{p},\theta_{\mathbf{p}}}$ as follows:

$$\mathcal{F}_{\mathbf{p},\theta_{\mathbf{p}}}(\mathcal{V}) = \mathbb{1}(\mathbf{V}_{\mathbf{p}}(\mathcal{V}) > \theta_{\mathbf{p}}), \tag{8}$$

where $V_{\mathbf{p}}$ is the response of pattern $\mathbf{p}$ on video $\mathcal{V}$ as in Eq. 6 and $\mathbb{1}(\cdot)$ is an identity function. To handle videos with different durations, a normalization weight is added to each frame so that all the video sequences are normalized to the same duration, denoted as N_T. Thus, $\theta_{\mathbf{p}}$ is the response threshold on the normalized sequences. The optimal $\theta_{\mathbf{p}}$ can be obtained by minimizing the following classification error:

$$\epsilon_{\mathbf{p}} = \min_{\theta_{\mathbf{p}}} \frac{1}{N_R} \sum_{i=1}^{N_R} \mathbb{1}(\mathcal{F}_{\mathbf{p},\theta_{\mathbf{p}}}(\mathcal{V}) \neq y_i), \tag{9}$$

where $y_i \in \{0,1\}$ is the label of the video. For simplicity, we denote $\mathcal{F}_\mathbf{p}$ as the orderlet classifier with parameter $\theta_\mathbf{p}$ obtained from solving Eq. 9.

All the size-2 orderlets, denoted as $\mathcal{T}_2$, can be sorted based on the classification error defined in Eq. 9. To discard redundant orderlets, we remove the orderlets which have large overlapping items with previous orderlets from the sorted list. This can make our orderlet pool more diverse. Only those orderlets at the top of the list, for example, $\epsilon_\mathbf{p} < \mu$, are kept for further processing, where μ is an error threshold. We denote this orderlet set as $\mathcal{T}_2'$. Given the size-2 orderlets in $\mathcal{T}_2'$, they can be easily extended to size-3 orderlet set $\mathcal{T}_3$ by adding one more element from $\Lambda^{(1)}$ to the end of size-2 orderlet. Similarly, based on Eq. 9, we select a subset of them with smaller classification error. This process will continue until it reaches size-L orderlet. By generating size-$(l+1)$ orderlets from discriminative size-l orderlets, it can save us a lot of computational cost since we do not need to enumerate all the size-$(l+1)$ orderlets, but only extend those size-l orderlets with smaller classification error.

We can apply the same algorithm to the other feature types as well. Note that for spatial coordinate in Eq. 2, we require that all the items in one orderlet should use the same coordinate type (x,y, or z). Thus the total number of candidates for size-2 orderlets is: $|\Lambda^{(2)}| \times (N_S - 1)$. In general, Algorithm 1 is a description of our orderlet discovery process. After the pattern discovery process, we have an orderlet pool $\mathcal{P}^{(f)}$ for each primitive feature f.

```
Algorithm 1 Orderlet Discovery
Input: Initial orderlet set $\mathcal{T}_2$, maximum orderlet size $L$
Output: Pattern Pool $\mathcal{P}^{(f)}$
 1: $\mathcal{P}^{(f)} := \emptyset$
 2: for $l := 2 \to L$ do
 3:    $\mathcal{T}_l' := \{\mathbf{p}_j | \mathbf{p}_j \in \mathcal{T}_l, \epsilon_{\mathbf{p}_j} < \mu\}$
 4:    $\mathcal{P}^{(f)} := \mathcal{P}^{(f)} \cup \mathcal{T}_l'$
 5:    if $l < L$ then
 6:       $\mathcal{T}_{l+1} := \emptyset$
 7:       for $\lambda^{(f)}_{i_{l+1}} \in \Lambda^{(f)} - O_{\mathbf{p}_j}$ where $\mathbf{p}_j \in \mathcal{T}_l'$ do
 8:          for $k := 1 \to l+1$ do
 9:             $\mathbf{p}^* := ([O_{\mathbf{p}_j}, \lambda^{(f)}_{i_{l+1}}], k)$
10:             $\mathcal{T}_{l+1} := \mathcal{T}_{l+1} \cup \mathbf{p}^*$
11:          end for
12:       end for
13:    end if
14: end for
```

3.5 Boosting Orderlets

As we have four types of primitive features, after mining the discriminative orderlet from each category, the combined orderlet pool is $\mathcal{P} = \mathcal{P}^{(1)} \cup \mathcal{P}^{(2)} \cup \mathcal{P}^{(3)} \cup \mathcal{P}^{(4)}$. Now we need to further select and combine the discovered orderlets for action classification. AdaBoosting [6] is employed here due to its good performance on feature selection and combination.

Given each orderlet $\mathbf{p} \in \mathcal{P}$, we have a corresponding weak classifier $\mathcal{F}_{\mathbf{p}}(\mathcal{V})$ as defined in Eq. 8. After the boosting stage, our final classifier can be computed as:

$$g(\mathcal{V}) = \mathbb{1}(\sum_{m=1}^{M} \alpha_m \mathbb{1}(\mathcal{F}_m(\mathcal{V}) = 1) > \sum_{m=1}^{M} \alpha_m \mathbb{1}(\mathcal{F}_m(\mathcal{V}) = 0)), \tag{10}$$

where M is the number of weak learners, $\mathcal{F}_m$ is the learned orderlet weak classifier, and α_m is the weight for the m-th weak classifier $\mathcal{F}_m$.

It is easy to extend our algorithm for multi-class action recognition. Suppose we have C categories of actions, for each category, a binary one-against-rest classifier is learnt. Then the testing video $\mathcal{V}$ is labeled as the category c^* with the maximum response:

$$c^* = \arg\max_c \sum_{m=1}^{M} \alpha_m^c (\mathbb{1}(\mathcal{F}_m^c(\mathcal{V}) = 1) - \mathbb{1}(\mathcal{F}_m^c(\mathcal{V}) = 0)), \tag{11}$$

where $\mathcal{F}_m^c, c = 1, 2, \cdots, C$, is the m-th weak classifier and α_m^c is its corresponding weight for the category c.

4 Online Action Recognition

Online action recognition performs real-time continuous prediction for on-going testing sequence. Let us first consider two-class online action recognition. Different from the sequence-level score in Eq. 10, we define the frame-level score as:

$$h(I_t) = \sum_{m=1}^{M} \alpha_m h_m(I_t), \tag{12}$$

where $h_m(I_t)$ is the response of orderlet $\mathbf{p}_m$ (the selected orderlet for the m-th boosting stage $\mathcal{F}^m(I_t)$) on frame I_t:

$$h_m(I_t) = \frac{N_T - \theta_{\mathbf{p}_m}}{N_T} \mathbb{1}(\mathbf{v}_{\mathbf{p}_m}(I^t) = 1) - \frac{\theta_{\mathbf{p}_m}}{N_T} \mathbb{1}(\mathbf{v}_{\mathbf{p}_m}(I^t) = 0), \tag{13}$$

where $\mathbf{v}_{\mathbf{p}_m}(I^t)$ is defined in Eq. 5. $\theta_{\mathbf{p}_m}$ and N_T are defined in Section 3.4. The weights $\frac{N_T - \theta_{\mathbf{p}_m}}{N_T}$ and $\frac{\theta_{\mathbf{p}_m}}{N_T}$ in Eq. 13 balance the positive and negative votes based on the response threshold $\theta_{\mathbf{p}_m}$.

As it is unreliable to make a decision based on a single frame, temporal smoothness is commonly applied to bring more robust result, e.g., by using a fixed-length window. However, it is difficult to determine the optimal window size since the action speed varies and different types of actions have different durations. We thus propose to use a smoothing window with adaptive temporal length. Since each frame votes a positive score for the target class and a negative score for the other types of actions, at current frame t, we can search backward for a window with the largest accumulated score. Following the maximum subarray search [23], we present an efficient forward sub-path search algorithm which can determine the best score for the current frame without performing a backward search for every frame. The idea is to maintain a best score $\mathcal{S}(\mathcal{V}^t)$ for each frame t:

$$\mathcal{S}(\mathcal{V}^t) = \max(\ 0, \mathcal{S}(\mathcal{V}^{t-1}) + h(I_t)\ ), \quad t > 1 \tag{14}$$

where $\mathcal{S}(\mathcal{V}^1) = h(I_1)$. If the best score is smaller than 0, it will be reset as 0, indicating the start of new action. Intuitively, $\mathcal{S}(\mathcal{V}^t) > 0$ means one action is continuing and $\mathcal{S}(\mathcal{V}^t) \leq 0$ means no action is happening at current frame. This can naturally split the testing sequence based on our score response defined by Eq. 14.

For multi-class online action recognition, denote $\mathcal{S}^c(\mathcal{V}^t)$ as the score for category c, then the category with largest response $c^* = \arg\max_c \mathcal{S}^c(\mathcal{V}^t)$ is the prediction label.

5 Experiments

5.1 Action Recognition on Online RGBD Action Dataset (ORGBD)

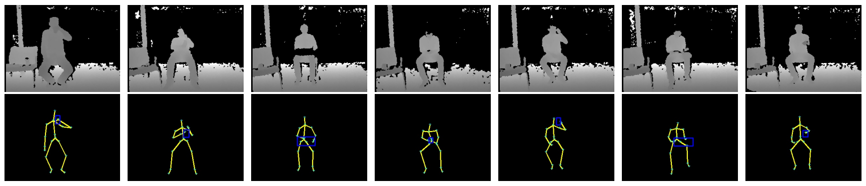

Fig. 2. Sample frames of our Human-Object Interaction dataset. The first row is from the depth stream and the second row is from the skeleton stream with the object position marked with blue rectangle. The seven columns refer to drinking, eating, using laptop, reading phone, picking up phone, reading book, and using remote, respectively.

As far as we know, there does not exist a benchmark dataset for cross-environment and online action recognition with depth sensors. To that end, we collect a new dataset which simulates the living room environment: "Online RGBD Action dataset (ORGBD)"[1]. There are seven types of actions that people often do in the living room: *drinking, eating, using laptop, picking up phone, reading phone (sending SMS), reading book*, and *using remote*. All these actions are human-object interactions. The bounding box of the object in each frame is manually labelled. The object location labels are used only for training. Fig. 2 gives an illustration of the seven action categories.

Three sets of depth sequences are collected by using a Kinect device. The first set, which is designed for action recognition in the same environment, contains

[1] The dataset can be downloaded from `http://research.microsoft.com/en-us/um/people/zliu/ActionRecoRsrc/default.htm`

16 subjects and each subject performs every action two times. The second set contains 8 new subjects which are recorded in different environments from the first one. This set is used for cross-environment action recognition. In the third set, each sequence consists of multiple unsegmented actions. This set is for real-time online action recognition.

Same-Environment Action Recognition We first evaluate our algorithm based on video sequences in the same environment: half-of the subjects in the first set are used for the training and the other half subjects are used for testing. 2-fold cross-validation is used and the mean value is reported in Table 1. It shows the significant improvement if multiple features are combined together compared with using a single feature. This verifies that our skeleton and object features are complementary. In addition to using boosting to select the discriminative orderlets, we also perform the experiment with linear SVM to learn the weights for each orderlet. Table 1 shows that boosting works better than linear SVM. Compared with the state-of-art algorithms [4, 17, 19, 27], our algorithm has obvious performance advantage.

Method	Accuracy
Pairwise joint distance only (Eq. 1)	0.633
Spatial coordinate only (Eq. 2)	0.544
Temporal variation only (Eq. 3)	0.455
Object feature only (Eq. 7)	0.464
All the features + Boosting	**0.714**
All the features + SVM	0.687
Skeleton + LoP [4]	0.660
DSTIP+DCSF [17]	0.617
EigenJoints [19]	0.491
Moving Pose [27]	0.384
All the features & Occlusion	0.546
EigenJoints [19] & Occlusion	0.169

Table 1. Comparison of recognition results on Human-Object Interaction Dataset.

Fig. 3 shows seven mined orderlets, one per action class, based on the pairwise joint distance. Each pair of joints is marked with the same color. The joint pair with red color refers to the minimum element index k which is defined in Eq. 4. For instance, for the drinking action, if the red color pair is of the smallest distance, then this orderlet will give a positive vote for the drinking action. The pattern statistics of our mined classifier in Eq. 10 is shown in Fig. 4. The diagram on the left shows the percentage of orderlets from each primitive feature type in the final classifier. The statistics of the orderlet size is shown in the right diagram of Fig 4. We can see that both skeleton and object orderlets have a strong effect on our final classifier and most of the orderlets have size 2 or 3.

Next we test the robustness of our algorithm against missing joints caused by occlusion or skeleton tracking module. With the Kinect device, it is difficult

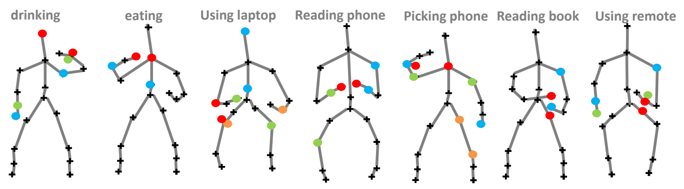

Fig. 3. Examples of mined orderlet based on pairwise joint distance. Different skeleton pairs are marked with different color.

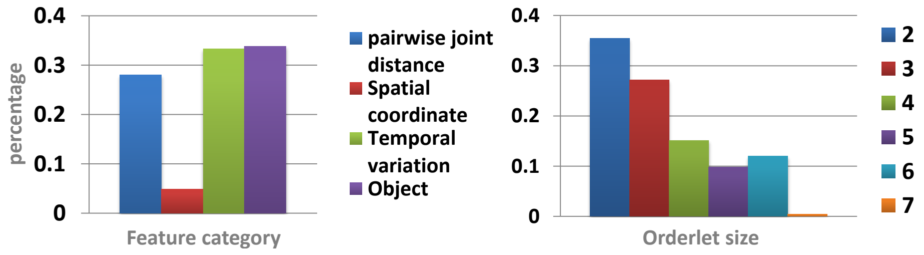

Fig. 4. Pattern statistics for our final classifier.

to capture partial occlusion data because the occlusion of a subset of the joints may cause the skeleton tracking to fail on all the joints. So we simulate the occlusion scenario by randomly selecting 4 skeleton joints and consider them as being occluded (setting their coordinates to be 0). The results are shown at the bottom of Table 1. We can see that our algorithm works much better than the global skeleton feature [19]. This is because most of the orderlets involve only a small number of joints.

Cross-Environment Action Recognition To test the robustness and generalization capability of our algorithm, we use the 8 new subjects from the second set of our human-object interaction dataset for testing. The 16 subjects in the first set are used for training. The recognition results are shown in Table 2. We can see that our algorithm is more robust than [4, 17, 19, 27] for cross-environment action recognition.

Method	Accuracy
All the features	**0.661**
Skeleton + LoP [4]	0.598
DSTIP+DCSF [17]	0.215
Eigenjoints [19]	0.357
Moving Pose [27]	0.285

Table 2. Comparison of cross-environment recognition results.

Action Prediction on Segmented Videos Action prediction [18] on segmented video sequences is tested to evaluate the latency of our algorithm. We follow the setting in Section 5.1. Fig. 5 shows the result of our algorithm. Obviously, our algorithm has significant advantages over DSTIP+DCSF [17] and Moving Pose [27] especially when the observation ratio is lower than 40%. To implement [17], 500 interest points are extracted from each depth sequence and clustered based on a vocabulary with 1000 words. From Fig. 5, we can see that our algorithm performs very well even when only 10% to 20% of the frames are observed. This is an indication of the effectiveness our object feature. When more frames are observed, the skeleton feature helps to improve the prediction results.

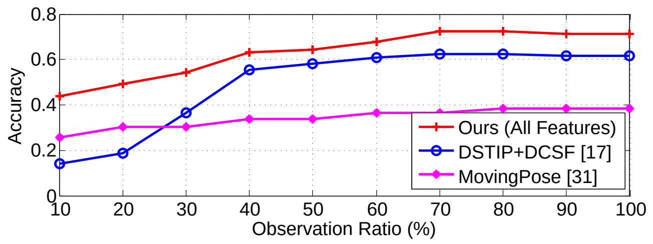

Fig. 5. Comparison of action prediction results.

Continuous Action Recognition Different from action prediction in previous subsection, which requires to know the start frame of the action, our algorithm can also be used for online action recognition on unsegmented video. We collect 36 unsegmented action sequences from 12 new subjects. There are three unsegmented sequences for each subject. Each sequence consists of multiple actions. Seven categories of human actions as well as the background action (none of the seven actions) are recorded continuously. The duration of these video sequences lasts from 30 seconds to 2 minutes. Among the 36 continuous sequences, there are 123 actions from the seven categories and the percentage of background frames (without any of the seven actions) is around 30%. For evaluation purpose, we manually label each frame of the video sequences but the boundary between two consecutive actions may not be very accurate since it is difficult to determine the boundary.

We train our model on the segmented data as described in Section 5.1 plus additional background action sequences. The 36 continuous videos are used for testing. Our algorithm can process around 25 frames per second with un-optimized code on a normal desktop PC.

Table 3 compares our algorithm with [17, 19, 27] based on frame-level accuracy, i.e., the percentage of frames which are correctly classified. To utilize [17] for online action recognition, we extract 3 DSTIPs in average from each frame and make a decision based on the histogram in a sliding window of 100 frames long. Similarly, [19, 27] are also modified to make a prediction based on a 100-frame sliding window. The results are shown in Table 3. We can see that our algorithm performs significantly better than the baselines [17, 19, 27] on the continuous action recognition.

Method	Accuracy
Our algorithm	**0.564**
DSTIP+DCSF [17]	0.321
EigenJoints [19]	0.236
Moving Pose [27]	0.50

Table 3. Comparison of online action recognition results based on frame-level accuracy.

Actually, there is a latency between the our prediction and the ground-truth. Fig. 6 gives a latency analysis of our algorithm. 61 actions out of 123 non-background actions, which have more than 50% overlap with ground-truth, are evaluated. Y-axis refers to the average ratio of the number of frames that have passed until our algorithm first gives a positive prediction over that action's duration. Clearly, our algorithm has a very short latency. Most of the time, it gives a correct prediction when less than 25% of the sequence is observed.

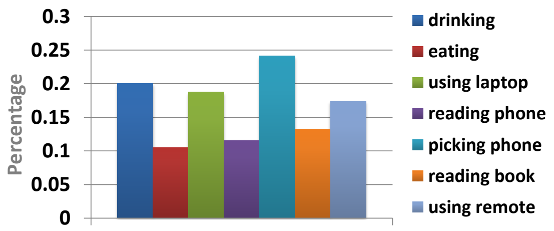

Fig. 6. Latency Analysis. Y-axis refers to the percentage of frames passed until our algorithm can make the positive prediction.

5.2 Action Recognition on MSR Daily Activity 3D Dataset

The MSR Daily Activity 3D dataset is collected in an indoor environment with sixteen categories: *drink, eat, read book, call cellphone, write on a paper, use laptop, use vacuum cleaner, cheer up, sit still, toss paper, play game, lie down on sofa, walk, play guitar, stand up and sit down*. There are 10 subjects and each subject performs each action twice, one in sitting pose and the other in standing pose.

Continuous Action Recognition To test continuous action recognition on MSR Daily Activity dataset, we use the videos from half of the subjects for training and the other half for testing. Similar to Section 5.1, frame-level accuracy is employed to evaluate the performance of action recognition based on all the frames till the current frame. As shown in Table 4, our algorithm obtains promising results compared with the baselines on the task of continuous action recognition.

Method	Accuracy
Our algorithm	**0.601**
DSTIP+DCSF [17]	0.246
EigenJoints [19]	0.470
Moving Pose [27]	0.452

Table 4. Comparison of continuous action recognition results on MSR Daily Activity Dataset. Frame-level accuracy is used for evaluating the performance.

Batch Action Recognition For the batch action recognition, we follow the standard evaluation setting: half of the subjects are used for training and the other half are used for testing. Table 5 shows the action recognition results. Since many of the categories in MSR Daily Activity dataset do not contain objects interacted with human, our algorithm is only based on skeleton feature. To make it fair for comparison, we mainly compare with the algorithms on skeleton feature [27] and [4] (skeleton only). Although our algorithm is not as good as [4] which is based on both skeleton and depth features on batch action recognition, [4] cannot be applied for the continuous action recognition due to the batch feature representation.

Method	Accuracy
Our skeleton Feature	0.738
Skeleton in [4]	0.68
Both Skeleton and Depth in [4]	0.857[2]
Moving Pose [27]	0.738

Table 5. Comparison of batch action recognition results on MSR Daily Activity dataset.

6 Conclusion

In this paper, we proposed a novel middle level representation, called orderlets, for recognizing human-object interactions. An orderlet captures the ordinal patterns among a group of low-level features. It can be applied to skeletons to encode inter-joint coordinations as well as depth maps to encode the object's shape information. An orderlet mining algorithm is presented to discover the most representative orderlets from an extremely large pool. A boosting technique is developed to combine the discriminative orderlets for action recognition. Experiments on cross-environment action recognition, occlusion handling, and online action recognition demonstrated the effectiveness of this new representation.

[2] This result (0.857) of [4] is obtained based on both the skeleton and depth streams while the other algorithms only rely on the skeleton features.

References

1. J. Shotton, A. 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.
2. O. Chum, J. Philbin, A. Zisserman "Near Duplicate Image Detection: min-Hash and tf-idf Weighting," in *BMVC*, 2008.
3. J. Yagnik, D. Strelow, D. Ross, R.S. Lin "The Power of Comparative Reasoning," in *ICCV*, 2011.
4. J. Wang, Z. Liu, Y. Wu, J. Yuan "Mining Actionlet Ensemble for Action Recognition with Depth Cameras," in *CVPR*, 2012.
5. J. Wang, Z. Liu, J. Chorowski, Z. Chen, Y. Wu "Robust 3D Action Recognition with Random Occupancy Patterns," in *ECCV*, 2012.
6. R. Schapire, "A brief introduction to boosting," in *IJCAI*, 1999.
7. S. Tang, X. Wang, T. Han, J. Keller, M. Skubic, S. Lao, Z. He, "Histogram of oriented normal vectors for object recognition with a depth sensor," in *ACCV*, 2012.
8. J. Liu, B. Kuipers, S. Savarese, "Recognizing Human Actions by Attributes," in *CVPR*, 2011.
9. C. Schuldt, I. Laptev, B. Caputo, "Recognizing human actions: a local svm approach," in *ICPR*, 2004.
10. G. Yu, J. Yuan, Z. Liu, "Unsupervised Random Forest Indexing for Fast Action Search," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
11. G. Yu, J. Yuan, Z. Liu, "Propagative Hough Voting for Human Activity Recognition," in *European Conference on Computer Vision*, 2012.
12. O. Oreifej, Z. Liu, "HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences," in *CVPR*, 2013.
13. I. Laptev, "On space-time interest points," in *IJCV*, vol. 64, no. 2-3, pp. 107–123, 2005.
14. P. Dollar, V. Rabaud, G. Cottrell, S. Belongiel, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005
15. H. Wang, A. Klaser, C. Schmid, C.L. Liu, "Action recognition by dense trajectories," in *CVPR*, 2011.
16. Y.G. Jiang, Q. Dai, X. Xue, W. Liu, C.W. Ngo, "Trajectory-based modeling of human actions with motion reference points," in *ECCV*, 2012.
17. L. Xia, J.K. Aggarwal, "Spatio-Temporal Depth Cuboid Similarity Feature for Activity Recognition Using Depth Camera," in *CVPR*, 2013.
18. M.S. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *ICCV*, 2011.
19. X. Yang, Y. Tian, "EigenJoints-based Action Recognition Using Naive-Bayes-Nearest-Neighbor," in *CVPRW*, 2012.
20. X. Yang, C. Zhang, Y. Tian, "Recognizing actions using depth motion maps-based histograms of oriented gradients," in *ACM Multimedia*, 2012.
21. H.S. Chen, H.T. Chen, Y.W. Chen, S.Y. Lee., "Human action recognition using star skeleton," in *ACM international workshop on Video surveillance and sensor networks*, 2006.
22. W. Li, Z. Zhang, Z. Liu, "Action recognition based on a bag of 3d points," in *CVPRW*, 2010.

23. J. Bentley, "Programming pearls: algorithm design techniques," in *Communications of the ACM*, Vol. 27(9):865C873, 1984.
24. Y. Zhu, W. Chen, G.D. Guo, "Fusing spatiotemporal features and joints for 3D action recognition," in *CVPRW*, 2013.
25. M. Hoai, F. DelaTorre, "Max-Margin Early Event Detectors," in *CVPR*, 2012.
26. B. Zhou, X. Wang, X. Tang, "Understanding Collective Crowd Behaviors: Learning a Mixture Model of Dynamic Pedestrian-Agents," in *CVPR*, 2012.
27. M. Zanfir, M. Leordeanu, C. Sminchisescu, "The Moving Pose : An Efficient 3D Kinematics Descriptor for Low-Latency Action Recognition and Detection," in *ICCV*, 2013.
28. G. Yu, A. Norberto, J. Yuan, Z. Liu, "Fast Action Detection via Discriminative Random Forest Voting and Top-K Subvolume Search," in *IEEE Transaction on Multimedia*, Vol.13, Issue. 3, pp. 507-517, 2011.
29. S. Sadanand, J.J. Corso, "Action bank: A high-level representation of activity in video," in *CVPR*, 2012.
30. C.Y. Chen, K. Grauman, "Efficient Activity Detection with Max-Subgraph Search," in *CVPR*, 2012.
31. A. Gupta, L.S. Davis, "Objects in Action: An Approach for Combining Action Understanding and Object Perception," in *CVPR*, 2007.
32. A. Jain, A. Gupta, M. Rodriguez, L.S. Davis, "Representing Videos using Mid-level Discriminative Patches," in *CVPR*, 2013.
33. D. Parikh, K. Grauman, "Relative attributes," in *ICCV*, 2011.