A Novel Hierarchical Framework for Human Action Recognition

Hongzhao Chen[a], Guijin Wang[a,*], Jing-Hao Xue[b], Li He[a]

[a]*Department of Electronic Engineering, Tsinghua University, Beijing 100084, China*
[b]*Department of Statistical Science, University College London, London WC1E 6BT, UK*

Abstract

In this paper, we propose a novel two-level hierarchical framework for three-dimensional (3D) skeleton-based action recognition, in order to tackle the challenges of high intra-class variance, movement speed variability and high computational costs of action recognition. In the first level, a new part-based clustering module is proposed. In this module, we introduce a part-based five-dimensional (5D) feature vector to explore the most relevant joints of body parts in each action sequence, upon which action sequences are automatically clustered and the high intra-class variance is mitigated. In the second level, there are two modules, motion feature extraction and action graphs. In the module of motion feature extraction, we utilize the cluster-relevant joints only and present a new statistical principle to decide the time scale of motion features, to reduce computational costs and adapt to variable movement speed. In the action graphs module, we exploit these 3D skeleton-based motion features to build action graphs, and devise a new score function

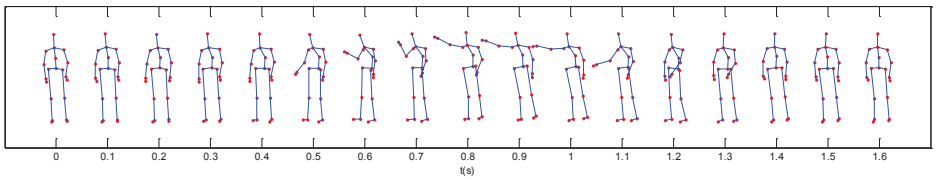
*Corresponding author. Tel.: +86-18911389502; Fax: +86-62770317
Email addresses: jordanchan1004@163.com (Hongzhao Chen), wangguijin@tsinghua.edu.cn (Guijin Wang), jinghao.xue@ucl.ac.uk (Jing-Hao Xue), hhappy06@gmail.com (Li He)

based on maximum-likelihood estimation for action graph-based recognition. Experiments on the Microsoft Research Action3D dataset and the University of Texas Kinect Action dataset demonstrate that our method is superior or at least comparable to other state-of-the-art methods, achieving 95.56% recognition rate on the former dataset and 95.96% on the latter one.
Keywords: action recognition, 3D skeleton, hierarchical framework, part-based, time scale, action graphs.

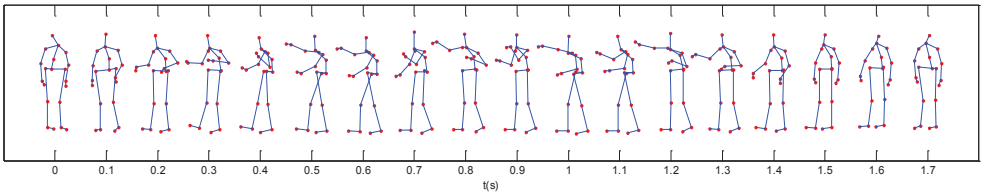
---

1. Introduction

Action recognition is an active research topic that focuses on labeling a motion sequence as one of the known actions. It can be widely applied in human-computer interaction, health care, video surveillance, etc. In order to achieve high accuracy and great robustness for real-world applications, an action recognition system has to overcome three challenges: high intra-class variance with low inter-class variance, variable movement speed, and high computational costs. As shown in Fig. 1, people may perform the same action of *Side Boxing* in quite different ways, by using one hand or two hands, leading to high intra-class variance. Meanwhile, people may also perform the same action with variable movement speed, as demonstrated in Fig. 2.



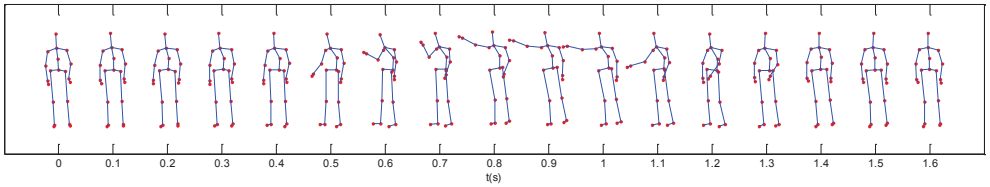
(a) Side Boxing with one hand



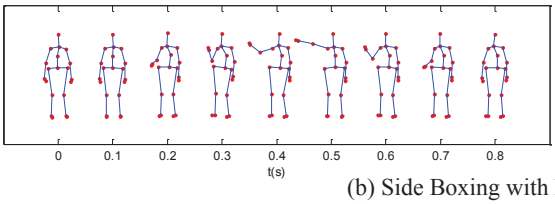
(b) Side Boxing with two hands

Figure 1: An illustrative example of high intra-class variance. Two panels present skeleton sequence diagrams of action *Side Boxing* sampled at 10fps.

Prior to 2010, many color image-based methods of action classification



(a) Side Boxing with low speed



(b) Side Boxing with high speed

Figure 2: An illustrative examples of variable movement speed. Two panels present skeleton sequence diagrams of action *Side Boxing* sampled at 10fps.

had been studied [1]. However, these methods have relatively low recognition accuracies and thus they are unable to be applied in real-world applications.

The situation has been much improved as technologies on depth imaging advance quickly [2–5]. Recent works of action recognition could be divided into two types, depth map-based methods [6–10] and 3D skeleton-based methods [11–22]. The former directly takes sequences of depth maps as input, while the latter utilizes 3D skeleton sequences inferred from depth maps. Fig. 3 shows the color image, depth image and 3D skeleton acquired from a Kinect sensor.

Depth map-based methods extract features from depth maps to describe the human poses and model the transition of poses. The widely-used features include sampled 3D points from silhouettes [6, 7], histograms of oriented gradients [8], histogram of oriented 4D normals [9], histogram of oriented principal components [10], etc. However, the extraction of these features is

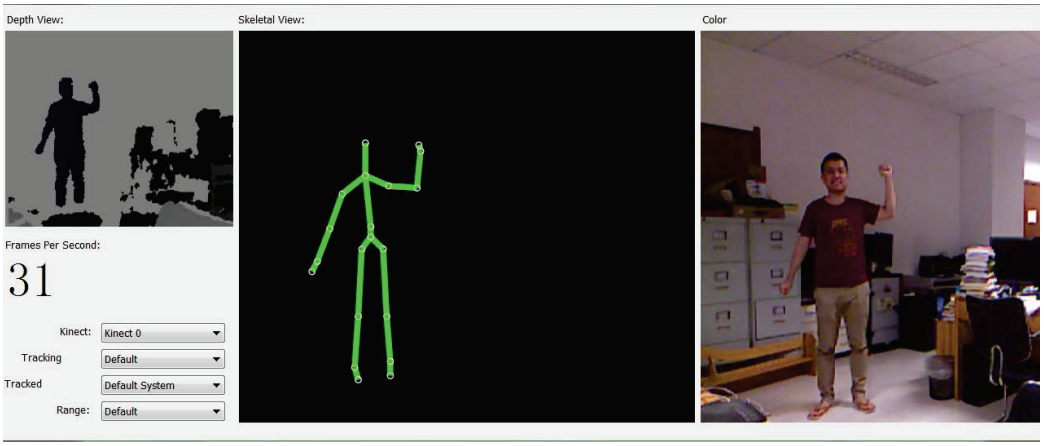


Figure 3: Depth image, skeleton and color image.

often time-consuming, making them hardly applicable in real-time scenarios.

In fact 3D human skeleton, which could be reliably estimated from depth maps in real time [23–25], is an efficient and concise surrogate to describe the human poses. In most 3D skeleton-based methods [11, 12, 14, 16, 18, 21], motion features are represented by pair-wise differences of joint positions within the current frame or between the current frame and the previous frames. Hence motion features extracted from 3D skeleton can efficiently model the action dynamics. Kapsouras *et al.* [21] further considered the time scale for motion features to fit various movement speeds. However, no principle has been supplied yet for how to determine the time scale. Some other histogram-based features are also proposed, like histograms of 3D joints [13], space time pose [17], histogram of oriented displacements [15], points in a lie group [19], etc.

Yang *et al.* [12] used a 3D skeleton-based method to achieve higher recognition rates than the depth map-based method [6], but in their method it was

still difficult to distinguish similar actions, such as *Draw X*, *Draw Tick* and *Draw Circle*. Mining techniques were adopted for finding relevant joints for each action [11, 14], which indeed improved the classification performance. However, mining techniques are computationally expensive and difficult to be expanded to new actions. Chen *et al.* [18] proposed a simple and effective hierarchical model to cluster similar actions and improved the accuracy of recognition. However, the hierarchical model therein was manually constructed, which limited the expansibility of the method. A new representation called SMIJ (sequence of the most informative joints) was proposed to select the most informative joints for performing an action [22], which was easy to interpret. However, this representation required segmenting each action sequence into several windows beforehand, which raised a difficult problem of choosing the temporal window size. Moreover, it performed poorly when the skeleton data were noisy or the actions were based on almost the same joints.

In this paper, in order to tackle the three challenges aforementioned, we propose a novel two-level hierarchical framework for 3D skeleton-based action recognition. The first level of the framework consists of a part-based clustering module. In this module, a part-based five-dimensional feature vector is introduced to explore the most relevant joints of body parts in each action sequence, upon which action sequences are clustered. Distinct sequences of the same action could be grouped into different clusters, enabling us to cope with the problem of high intra-class variance. Hence this module groups similar actions together and divides the recognition task for various actions into several smaller and simpler tasks, which can significantly improve the

final performance (for example by more than 10% in our first experiment). In the second level of the framework, there are two modules, motion feature extraction and action graphs. For each cluster, only the relevant joints obtained from the first level are utilized for motion feature extraction, which not only enhances the validity of the extracted features but also reduces the computational costs remarkably. We also investigate and derive a statistical principle for determining the time scale of motion features to deal with the problem of variable movement speed. After motion feature extraction, we apply action graphs to these 3D skeleton-based motion features to finally classify actions. Experiments on the Microsoft Research Action3D dataset and the UTKinect-Action dataset show that our method is noticeably superior or at least comparable to other state-of-the-art methods, achieving recognition rates of 95.56% and 95.96%, respectively on the two datasets.

The remainder of this paper is organized as follows. In Section 2 the proposed hierarchical framework is described. Details of the three key modules, part-based clustering, motion feature extraction and action graphs, are followed in Section 3. Experimental performance of our method is demonstrated in Section 4. Section 5 concludes our work and discusses the future work.

2. Hierarchical Framework

As shown in Fig. 4, our hierarchical framework for 3D skeleton-based action recognition consists of three modules: part-based clustering, motion feature extraction and action graphs classification. For an action sequence, the hierarchical framework first decides its cluster. Motion features are then

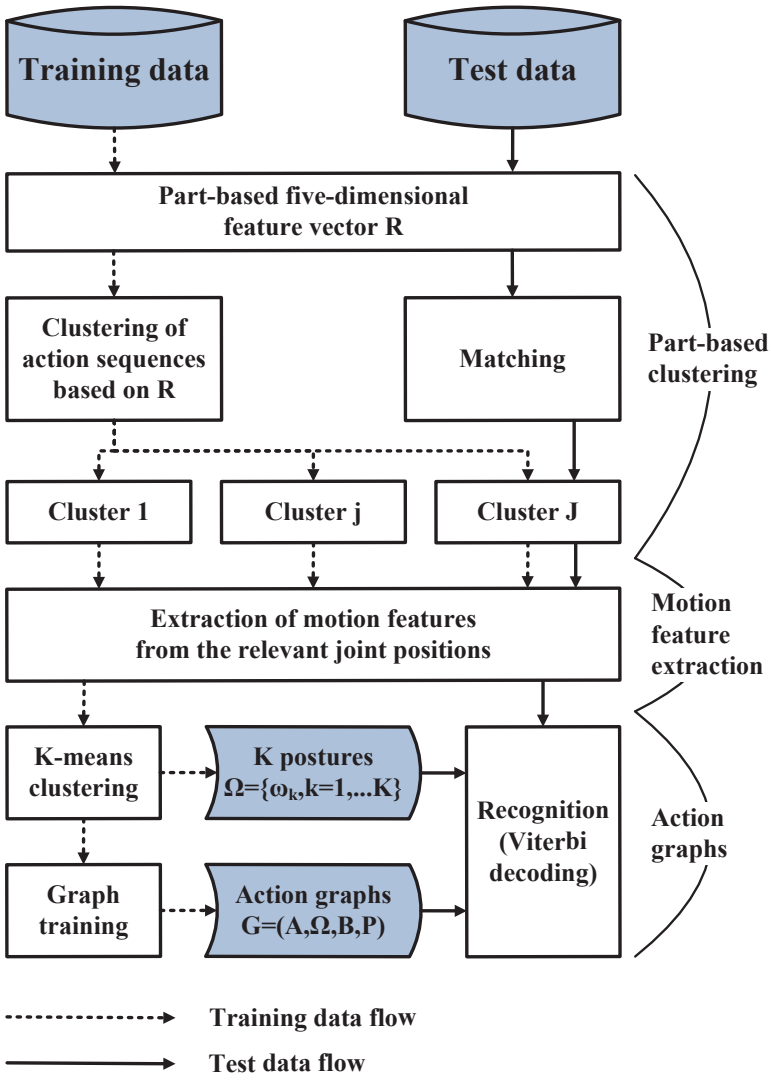


Figure 4: The hierarchical framework of our proposed method.

extracted from the relative joints. In the final classification, we apply the Viterbi decoding to the action graphs. Unlike [18], the hierarchical framework here is automatically learned from data during the training procedure without manual intervention. Correspondingly, our framework has three main characteristics.

Firstly, a part-based five-dimensional feature vector is introduced to explore the most relevant joints of body parts in each action sequence. Then action sequences are clustered to construct the first level of the hierarchical model. More details will be described in Section 3.1. The part-based clustering module groups similar actions together and partitions the recognition task for various actions into several smaller and simpler tasks, which could significantly improve the final performance, as verified in Section 4.3.1.

Secondly, for each cluster, only the relevant joints are utilized for motion feature extraction. Since irrelevant joints provide little information for classification, motion features of the relevant joints can not only enhance the validity of the extracted features but also reduce the computational costs. We extract motion features with higher order and time scale (see Section 3.2), which improves feature representability and thus achieves higher accuracy as demonstrated in Section 4.3.2. We further investigate how to choose the time scale for any specified dataset.

Thirdly, we apply action graphs for classification. Unlike [6], our action graphs are applied to skeleton-based motion features. Postures are obtained via K-means clustering, upon which action graphs are trained. From maximum likelihood estimation, we derive a new score function for the Viterbi decoding. We also investigate action graphs for early detection of actions,

which will be discussed in Section 4.3.3.

3. Modules

3.1. Part-based Clustering

The joints of a human body could be divided into several parts, as actions are only related to certain parts of the body. Given the relevance of parts for each action sequence, actions that are relevant to different joints could be discriminated easily. So we could cluster action sequences with same relevant parts together and divide the recognition task into several simpler tasks in a cluster. In our method, a part-based five-dimensional feature vector is defined to explore the most relevant joints to body parts in each action sequence, then action sequences are clustered by using these features to construct the first level of the hierarchical framework.

We define a body part as a set of joints close to each other. For clarity and simplicity, here we assume that the number of joints involved is 20 according to the Kinect sensor. Other situations with different numbers of joints could be adapted without difficulty. Here five body parts and the joints relevant to each body part are defined as

$$
\begin{aligned}
O_1 = LUE = &\{left\ shoulder(1), left\ elbow(2),\\
&left\ wrist(3), left\ hand(4)\};\\
O_2 = RUE = &\{right\ shoulder(5), right\ elbow(6),\\
&right\ wrist(7), right\ hand(8)\};\\
O_3 = LLE = &\{left\ hip(9), left\ knee(10),\\
&left\ ankle(11), left\ foot(12)\};\\
O_4 = RLE = &\{right\ hip(13), right\ knee(14),\\
&right\ ankle(15), right\ foot(16)\};\\
O_5 = TRS = &\{head(17), shoulder\ center(18),\\
&spine(19), hip\ center(20)\}.
\end{aligned}
$$

To measure the relevance of the five body parts to an action sequence, we construct a part-based five-dimensional feature vector $R = [R_1, \ldots, R_5]$.

Given a T-frame sequence of joints positions $X = \{X^1, \ldots, X^T\}$, where $X^t = [x_1^t, \ldots, x_{20}^t]$ in which x_i^t is the 3D coordinate of the ith joint in the tth frame, the variance vector $V = [V_1, \ldots, V_{20}]$ of the joints in the sequence is calculated as

$$
V_i = \sum_{t=1}^{T} \|x_i^t - \bar{x}_i\|^2 \ ,
$$

where the mean coordinate $\bar{x}_i = \frac{1}{T}\sum_{t=1}^{T} x_i^t$. For incomplete skeletons, the corresponding entries V_i can be simply set as zero for the missing joints. Then the feature vector R, which represents the relevance of body parts to

an action sequence, is defined as:

$$R_j = 1_{\{(\hat{R}_j / \max_{l\in\{1,\ldots,5\}} \hat{R}_l) > \eta\}}, \text{where } \hat{R}_j = \sum_{i\in O_j} V_i \ . \tag{1}$$

Here $1_{\{\cdot\}}$ is the indicator function valued in $\{0, 1\}$; the threshold η can be decided by cross-validation using the training set.

Then action sequences can be clustered based on R, which automatically constructs the first level of the hierarchical framework during the training phase. A practical example of part-based clustering is shown in Fig. 5. As different people may perform the same action in different ways (actions with large intra-class variances), distinct sequences of the same action are allowed to be grouped into different clusters. For example, one person may perform action *Side Boxing* with two hands while another person may do it with only one hand, which means that R for these two sequences are different so that they may belong to different clusters. Allowing distinct sequences of the same action to be grouped into different clusters improves the performance, as verified in Section 4.3.1.

Our method is also readily expansible for adding new actions. To add a new action to the dataset, we could simply calculate R for sequences of this new action, then join them to proper clusters or create a new cluster, and finally retrain the action graphs for the actions of the affected clusters.

3.2. Motion Feature Extraction

For each cluster, we extract motion features F^t, $t = 1, \ldots, T$, from the relevant joints $\hat{X}^t = \{x_i^t \mid i \in O_j, R_j = 1, j = 1, \ldots, 5\}$ only. For example, assume $R = [1, 1, 0, 0, 0]$ for a cluster, which means that the actions in this cluster are mainly relative to the two hands. Then the relevant joints

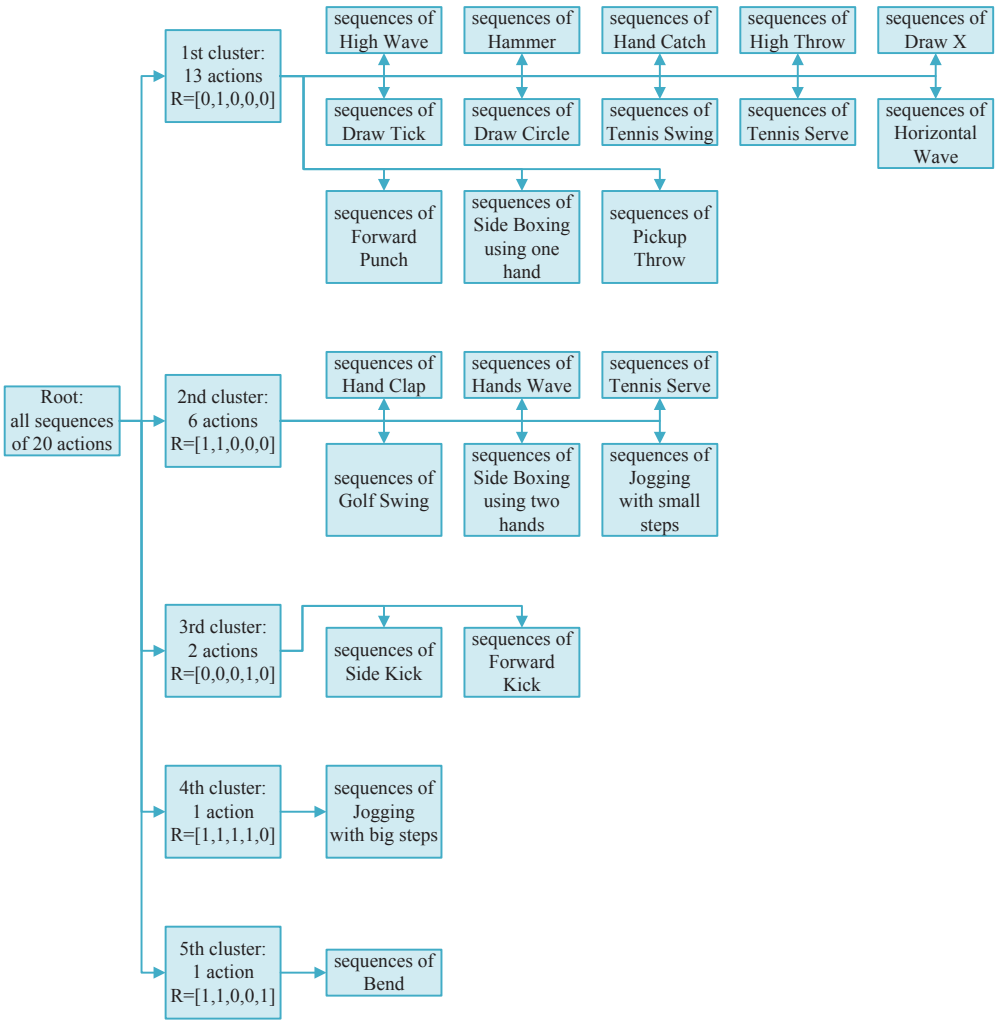


Figure 5: The part-based clustering result obtained from the 20 actions.

$\hat{X}^t$ are the joints of $O_1(LUE)$ and $O_2(RUE)$, i.e., $\hat{X}^t = \{x_1^t, x_2^t, \ldots, x_8^t\}$. As we extract motion features only from relevant joints, the dimension of $\hat{X}^t$ is reduced. Since joints of irrelevant parts provide little information for identification of actions, the motion features extracted from only the joints of relevant parts can not only enhance the validity of the features but also greatly reduce the computational costs. The procedure of motion feature extraction is described as follow.

Similarly to previous works [12, 21], we compute pair-wise differences of joint positions between specific frames to obtain motion features. These features consist of three components: static pose SP, dynamic pose DP and offset pose OP. The static pose represents the static state at the current frame, the dynamic pose represents the instant motion, and the offset pose represents the offset from the initial state.

In [12], the higher order terms (position differences of different joints) were utilized to reduce noise but without considering the time scale. In contrast, [21] considered the time scale but ignored the higher order terms. In this paper, we combine them together, which means: We calculate SP, OP and DP with higher order terms as in [12], while for DP, to take the time scale into consideration, we calculate it with three previous frames (1, 5, 10 frames) before the current frame, respectively. This combination aims to capture the dynamics in various time scales and to reduce noise at the same time, which could enhance representability of the feature. We further investigate and devise a statistical principle for choosing the time scale, i.e., the number of previous frames used. It will be demonstrated that, given a dataset, using the time scale up to the standard deviations of sequence

lengths is highly possible to offer an optimal performance. More details could be found in Section 4.3.2.

To extract the motion features for the tth frame of a T-frame sequence, we calculate the three components from only the relevant joint positions $\hat{X}^t = \{\hat{x}_m^t \mid m = 1, \ldots, M\}$, where M is the number of relevant joints:

$$SP^t = \{\hat{x}_i^t - \hat{x}_j^t \mid i, j = 1, \ldots, M; i \neq j\} \ ,$$

$$DP^t = \{\hat{x}_i^t - \hat{x}_j^{t-s} \mid i, j = 1, \ldots, M; s = 1, 5, 10\} \ ,$$

$$OP^t = \{\hat{x}_i^t - \hat{x}_j^1 \mid i, j = 1, \ldots, M\} \ .$$

Then we concatenate the three components to obtain the motion features $F_{ori}^t = [SP^t, DP^t, OP^t]$. After that, we apply principle component analysis (PCA) to reduce the dimension of the features and reach the final motion features

$$F^t = W_{opt}(F_{ori}^t - \mu) \ ,$$

where W_{opt} is the optimal projection matrix and μ is the mean of all F_{ori}^t. Note that here PCA is utilized to reduce computational costs and the resulting motion features are later fed into action graphs for classification rather than directly used for classification. Thus, supervised dimension-reduction techniques like linear discriminant analysis (LDA) are not suitable here. Denote the dimension of the final motion features F^t as L. The impact of L on recognition performance will be investigated by experiments in Section 4.4.

3.3. Action Graphs

Many classification methods have been introduced to action recognition, among which action graphs [26] are an effective method to explicitly model the action dynamics. Action graphs were applied to depth map-based action recognition in [6]. Unlike [6], here we apply action graphs to 3D skeleton-based features and derive a new score function for recognition.

An action can be represented by transitions of several postures, here a posture means the similar motion features for the frames in a salient state. Thus we could model an action as a weighted directed graph, whose nodes represent the postures of the action and whose edges represent the transitional probabilities between two postures.

Fig. 6 is a sketch of two action graphs. Action *High Throw* consists of three postures, ω_1 =*hand lifting*, ω_2 =*throwing* and ω_3 =*hand putting down*. The notation $p_n[i, j]$ near the arrows represent the transitional probabilities between postures ω_i and ω_j. In most cases of action *High Throw*, it starts from *hand lifting*, transits to *throwing* and ends at *hand putting down*. It is also quite possible that two consecutive frames stay at the same posture, which is represented as a self-loop edge in the action graph. Situations are similar for action *High Wave*, which shares two postures ω_1 and ω_3 with the former action and has another two new postures ω_4 =*wave slightly* and ω_5 =*wave substantially*. Nevertheless, because of high intra-class variance, there are two or more possible paths (e.g. 1-4-3 or 1-5-3) for the same action.

As actions often share postures, we could obtain postures by clustering the motion features of all frames from all training samples via an algorithm like K-means clustering, with each cluster center representing a posture.

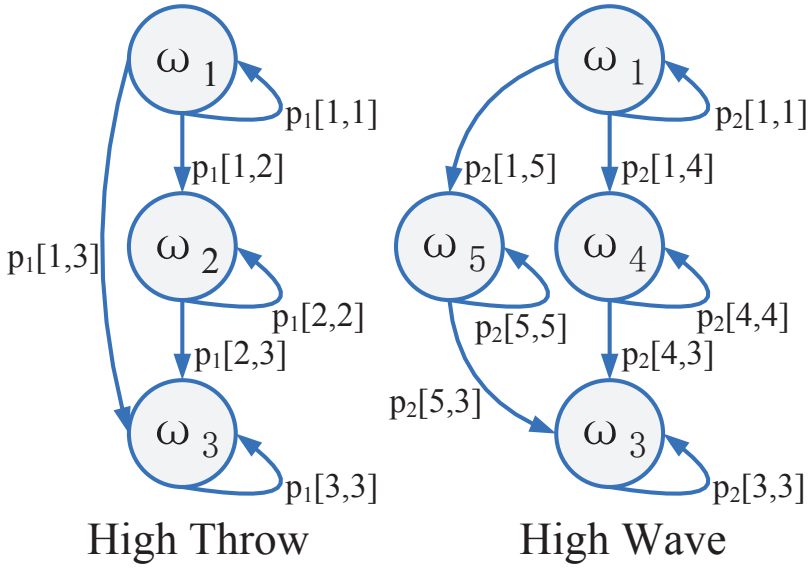


Figure 6: A sketch of two action graphs. In the sketch, each ω_i represents a posture and the notation $p_n[i, j]$ near the arrows represent the transitional probabilities between postures ω_i and ω_j for the nth action. This sketch intends to visualize the relationship of postures, actions and transitional probabilities.

Consider an action set of N actions $A = \{A_n\}_{n=1,\ldots,N}$ that contains K postures $\Omega = \{\omega_k\}_{k=1,\ldots,K}$, we could model them as a set of weighted directed graphs, which can be represented by a quadruplet $G = (A, \Omega, B, P)$, where

$$A = \{A_n\}_{n=1,\ldots,N} \ ,$$
$$\Omega = \{\omega_k\}_{k=1,\ldots,K} \ ,$$
$$B = \{B_n\}_{n=1,\ldots,N} \ ,$$
$$P = \{P_k\}_{k=1,\ldots,K} \ ,$$

in which $B_n = \{p(\omega_j|\omega_i, A_n)\}_{i,j=1,\ldots,K}$, for $n = 1, \ldots, N$, is the transitional probability matrix of the nth action A_n, and $P_k(F^t) = p(F^t|\omega_k)$ is the conditional probability of an observation F^t to be generated from the node ω_k. We assume that the distribution of the observations for a node can be approximated by an isotropic normal distribution: $P_k(F^t) = p(F^t|\omega_k) = \frac{1}{(2\pi)^{L/2}\sigma^L} \exp(-\frac{1}{2\sigma^2}\|F^t - \omega_k\|^2)$. Here σ is the standard deviation for all L dimensions. Such action graphs G can be trained as described in [26].

For classification, we could apply maximum likelihood estimation. Given an action sequence of T frames, let F^t be the final motion feature vector for the tth frame, and $F = \{F^t\}_{t=1,\ldots,T}$ be the motion feature sequence. The posture sequence corresponding to F is denoted by $S = \{S^t\}_{t=1,\ldots,T}$, where $S^t \in \Omega$ for all t. The recognition of the most likely action A^* that generates the observation F can be then formulated as a maximum likelihood

estimation:

$$
\begin{aligned}
A^* &= \underset{A_n \in A, S \in \Omega^T}{\arg\max}\ p(F, S|A_n) \\
&= \underset{A_n \in A, S \in \Omega^T}{\arg\max}\ \left[p(S|A_n)\, p(F|S, A_n)\right] \\
&= \underset{A_n \in A, S \in \Omega^T}{\arg\max}\ \left[p(S^1, \ldots, S^T|A_n)\, p(F^1, \ldots, F^T|S^1, \ldots, S^T, A_n)\right] \ .
\end{aligned} \tag{2}
$$

Assume 1) F is statistically independent of A_n given S, 2) F^t is statistically dependent only on S^t, and 3) S^t is a Markov chain in an action, i.e., S^t only depends on its previous state S^{t-1}. We can further simplify (2) as

$$
\begin{aligned}
A^* &= \underset{A_n \in A, S \in \Omega^T}{\arg\max} \prod_{t=1}^{T} \left[p(S^t|S^{t-1}, A_n)\, p(F^t|S^t)\right] \\
&= \underset{A_n \in A, S \in \Omega^T}{\arg\max} \sum_{t=1}^{T} \left[\log(p(S^t|S^{t-1}, A_n)) + \log(p(F^t|S^t))\right] \ .
\end{aligned} \tag{3}
$$

To solve (3), we adopt the Action-Specific Viterbi Decoding(ASVD) method [26] and derive a new score function:

$$
\begin{aligned}
& Score(A_n) \\
&= \max_{S \in \Omega^T} \sum_{t=1}^{T} \left[\log(p(S^t|S^{t-1}, A_n)) + \log(p(F^t|S^t))\right] \\
&= \max_{I \in \{1,\ldots,K\}^T} \sum_{t=1}^{T} \left[\log(p(\omega_{I^t}|\omega_{I^{t-1}}, A_n)) + \log(p(F^t|\omega_{I^t}))\right] \\
&= \max_{I \in \{1,\ldots,K\}^T} \sum_{t=1}^{T} \left[\log(B_n[I^{t-1}, I^t]) + \log(P_{I^t}(F^t))\right] \\
&= \max_{I \in \{1,\ldots,K\}^T} \sum_{t=1}^{T} \left[\log(B_n[I^{t-1}, I^t]) - C\|F^t - \omega_{I^t}\|^2\right] \ ,
\end{aligned} \tag{4}
$$

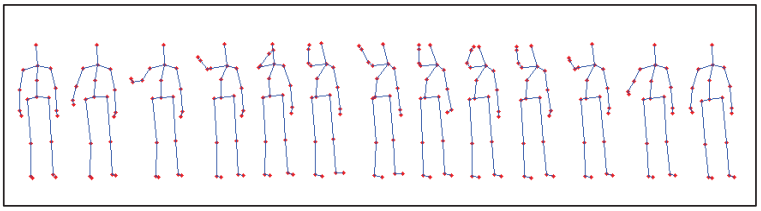
and

$$
A^* = \underset{A_n}{\arg\max}\, Score(A_n) \ . \tag{5}
$$

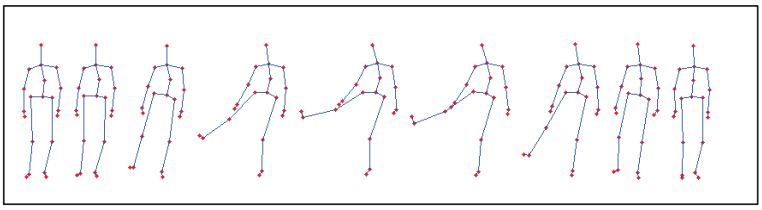
In (4), $I = [I^1, \ldots, I^T]$ is a sequence of numbers, where $I^t \in \{1, \ldots, K\}$, $\omega_{I^t} = S^t$, and $C = \frac{1}{2\sigma^2}$, which could be optimized by cross-validation using the training set if σ is hard to be reliably estimated.

4. Experiments and Discussion

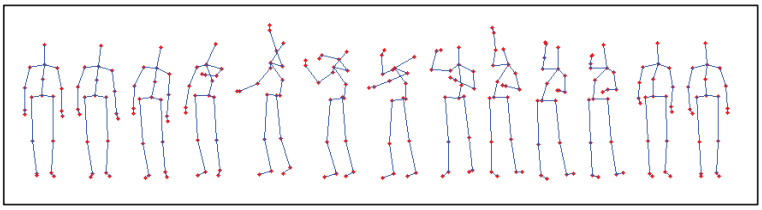
4.1. Datasets



(a) High Wave



(b) Side Kick



(c) Tennis Serve

Figure 7: Sample frames of 20-joints skeleton for actions of (a) High Wave, (b) Side Kick and (c) Tennis Serve from the MSRAction3D dataset.

MSRAction3D dataset [6]**:** The Microsoft Research Action3D dataset

("MSRAction3D" for short) consists of 20 actions of 10 subjects, each action with 2 or 3 repetitions. These actions are mainly interactions with console in video games. As shown in Fig. 7, actions in this dataset capture a variety of motions related to arms, legs, torso, and their combinations. Meanwhile, the skeleton positions in this dataset are quite noisy. Hence, experiments on this dataset were widely adopted to test the accuracy and robustness of recognition methods for various actions. In previous works, it was used in two ways, either as one dataset containing all actions [7, 9–11, 14, 19–21] or by division into three subsets [6, 8, 12, 13, 15–19, 21].

UTKinect-Action dataset [13]**:** The University of Texas Kinect Action dataset ("UTKinect-Action" for short) consists of 10 actions of 10 different persons, each action with 1 or 2 repetitions. These actions, including *walk, sit down, stand up, pick up, carry, throw, push, pull, wave* and *clap hands*, are mainly obtained from daily life. One of the challenges of this dataset is that one of the persons is left-handed. Meanwhile, the lengths of samples from this dataset vary in a wide range. It was used for evaluation in several previous works [13, 16, 17, 19].

4.2. Experiments and Results

In all experiments, we use the same hardware setup: Intel Core i7-4790 CPU @3.6GHz, 24GB RAM. Our method could run at about 30fps for test.

In all experiments, C in (4) and η in (1) are tuned by cross-validation using the training set, which follows the procedure below: (i) set a grid of values of C and η, divide the training set into two halves; (ii) train on the first half with different combinations of C and η, get accuracies by testing on the second half; (iii) repeat (ii) but exchange the two halves, average the

accuracies from (ii) and (iii); and (iv) find the values of C and η that produce the highest accuracy.

4.2.1. MSRAction3D

We follow the two types of experiments, as with the previous works. Firstly, the whole dataset is used to verify the performance of our method on a large number of actions. Secondly, three subsets of this dataset are tested to confirm the applicability of the method in various situations.

In the first type of experiments, the entire 20 actions with 557 sequences are applied as with [9]. We conduct the cross-subject test, where sequences of half the subjects are used for training and the rest for testing. We repeat the experiment 252 times for different folds of 10 subjects as with [9]. The performance, including the best result, the worst result and the average result, are listed and compared in Table 1. The confusion matrix of the best result is displayed in Fig. 8. In the confusion matrix, the vertical coordinate (y) represents the true label of an action sequence and the horizontal coordinate (x) represents the recognition result. The value at the (x, y) coordinate of the matrix represents the ratio of action y recognized as action x. As shown in Table 1, our method outperforms other methods in terms of the average result and/or the best result. From the confusion matrix of the best result in Fig. 8, we can see that 15 out of 20 actions achieve 100% accuracy, which is perfect considering the noise of skeleton collected by Kinect and the huge intra-class variance among different subjects. Note that the action *Hand Catch* gets the lowest accuracy 50%, mainly because it is similar to other two actions *High Wave* and *High Throw*.

In the second type of experiments, as with [6] all the 567 sequences of the

Table 1: Recognition rates of the first experiment on MSRAction3D. In the table, we present the best result, the worst result and the average result for 252 different folds of 10 subjects.

Method	Avg ± Std	Best	Worst
Ours	87.05 ± 3.75	**95.56**	74.39
Random Occupancy Patterns [7]	-	86.50	-
Actionlet Ensemble [11]	-	88.2	-
HON4D+D_{disc} [9]	82.15 ± 4.18	88.89	-
Spatial and Temporal Part Sets [14]	-	90.22	-
HOPC of 3D Pointclouds [10]	86.49 ± 2.28	92.39	74.36
Points in a Lie Group [19]	-	89.48	-
Histograms of Action Poses + DTW [20]	-	90.56	-
Dynemes and Forward Differences [21]	-	91.94	-

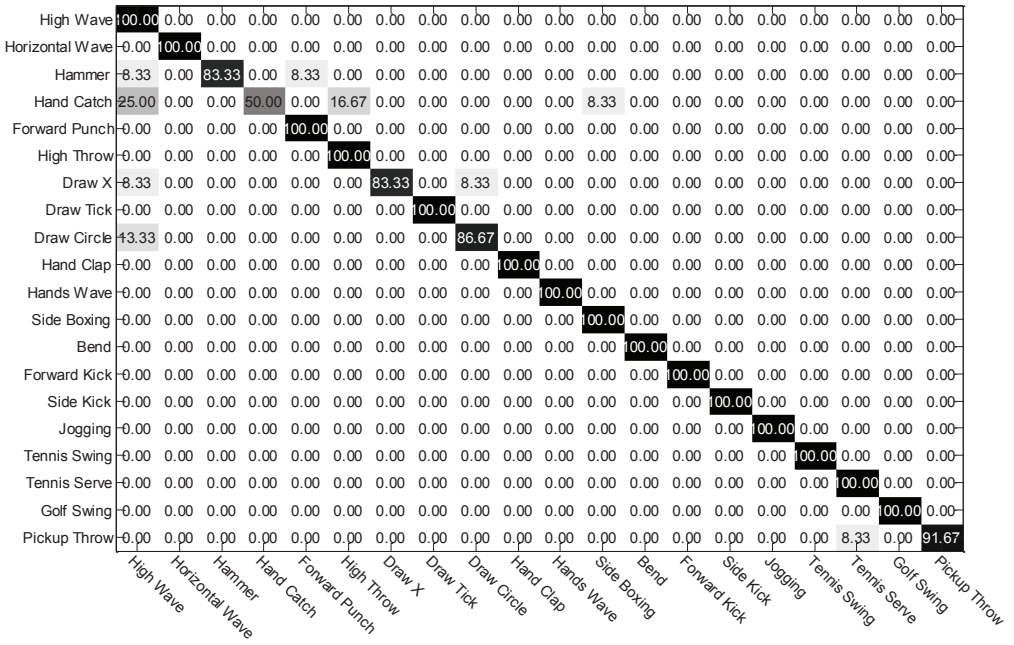


Figure 8: Confusion matrix of our method on MSRAction3D with the part-based clustering. In the confusion matrix, the vertical coordinate (y) represents the true label of an action sequence and the horizontal coordinate (x) represents the recognition result. The value at the (x, y) coordinate of the matrix represents the ratio of action y recognized as action x.

dataset are split into three subsets, each with eight actions. Evaluation for each subset is done independently. Table 2 shows the three subsets in this experiment. These three subsets have different aims: AS1 and AS2 contain similar actions to verify a method's ability to discriminate similar movements, while AS3 groups complex actions together to evaluate the versatility of a method. The overall recognition rate is calculated by averaging the results over subsets. We also use the cross-subject test for this experiment and repeat it for different folds of subjects. The results are shown in Table 3, from which we can observe that our method outperforms the state-of-the-art method [21] by 2.5% in terms of the best result.

Table 2: The three subsets of actions for the second experiment on MSRAction3D.

Action Set 1 (AS1)	Action Set 2 (AS2)	Action Set 3 (AS3)
Horizontal Wave (HoW)	High Wave (HiW)	High Throw (HT)
Hammer (H)	Hand Catch (HC)	Forward Kick (FK)
Forward Punch (FP)	Draw X (DX)	Side Kick (SK)
High Throw (HT)	Draw Tick (DT)	Jogging (J)
Hand Clap (HC)	Draw Circle (DC)	Tennis Swing (TSw)
Bend (B)	Hands Wave (HW)	Tennis Serve (TSr)
Tennis Serve (TSr)	Forward Kick (FK)	Golf Swing (GS)
Pickup Throw (PT)	Side Boxing (SB)	Pickup Throw (PT)

Table 3: Recognition rates of the second experiment on MSRAction3D.

Method	Avg ± Std	Best	Worst
Ours	88.7 ± 3.6	**96.1**	78.0
Bag of 3D Points [6]	-	74.7	-
Histograms of 3D Joints [13]	-	78.97	-
EigenJoints [12]	-	82.3	-
EigenJoints + Hierarchical [18]	-	90.3	-
Histograms of Oriented Displacements [15]	-	91.26	-
Random Forests [16]	-	94.3	-
Space-Time Pose [17]	-	92.77	-
Points in a Lie Group [19]	-	92.46	-
Dynemes and Forward Differences [21]	-	93.6	-

4.2.2. UTKinect-Action

We follow the challenging cross-subject test setting of [16, 19] instead of leave-one-out-cross-validation (LOOCV) setting of [13, 17]. In this setting, half of the subjects are used for training while the remaining for testing. The performances on this dataset are compared in Table 4. We can find out that our method, comparable to the method of Vemulapalli *et al.* [19], is much better than other methods. The confusion matrix of our method is shown in Fig.9, from which we can observe that all 10 actions achieve accuracy higher than 90%, 6 out of which achieve 100% accuracy.

Table 4: Recognition rates on UTKinect-Action.

Method	Recognition rate
Ours	95.96
Histograms of 3D Joints [13]	90.92
Random Forest [16]	91.9
Space-Time Pose [17]	91.5
Points in a Lie Group [19]	**97.08**

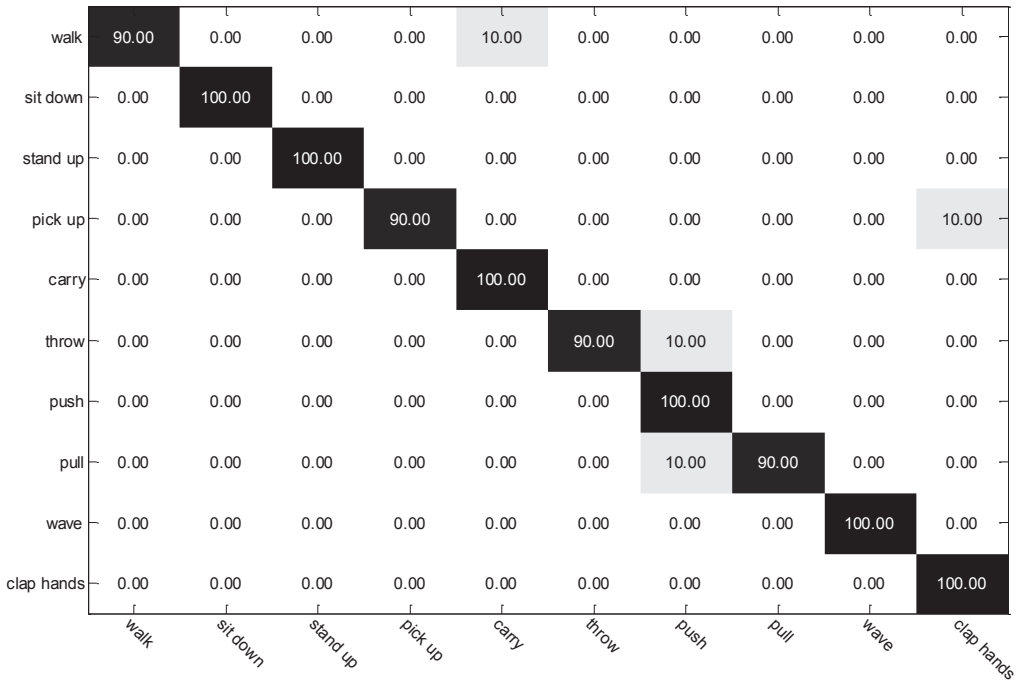


Figure 9: Confusion matrix of our method on UTKinect-Action.

4.3. Module Analysis

We carry out further experiments to analyze the effects of the three modules. All these experiments follow the same setup as the first type of experiments on MSRAction3D aforementioned, i.e. the entire 20 actions being used, cross-subject test and repeating for different folds of subjects.

4.3.1. Part-based Clustering

We perform the first type of experiments on MSRAction3D to verify the part-based clustering. The results are shown in Table 5. In terms of the average result, the part-based clustering significantly improves the recognition rate by about 14%. The improvement is mainly due to two merits of the part-based clustering: 1) it automatically divides distinct sequences of the same action into more than one cluster, and 2) it provides the relevant joints for motion feature extraction to help distinguish similar actions in the same cluster. Fig. 5 shows the part-based clustering result obtained from the 20 actions.

For the first merit, again take action *Side Boxing* as an example. Somebody may do it with one hand while others may prefer to use two hands, which results in a large intra-class variance (see Fig. 1). Without the part-based clustering, it only achieves 86.7% recognition accuracy. If we restrict that the sequences of the same action could only be grouped into a single cluster, it is difficult to decide which cluster, the cluster of using one hand or the cluster of using two hands, is correct for *Side Boxing*. Using either one may lead to misclassifying the other kind of sequences, which will result in a low recognition accuracy of 70%, even lower than the result without the part-based clustering. In contrast, allowing them to appear in different

clusters (see Fig. 5) can solve the problem, which leads to 100% recognition accuracy for this action in our experiment. A similar case is with action *Jogging*.

For the second merit, take actions *Hammer*, *High Throw* and *Draw Circle* as an example. Without the part-based clustering, action *Hammer* is highly confused with *High Throw* or *Draw Circle* and only 25% are recognized correctly. As these actions are similar movements using only the right hand, motion features extracted from all joints of body would contain much irrelevant information as noise, resulting in a low recognition accuracy. With the part-based clustering, they are grouped into the same cluster and only the joints of right up extreme (RUE) are used for motion feature extraction. These motion features are the most discriminative with the irrelevant noise dropped out, which ultimately improves the recognition rate of *Hammer* up to 83.3%. Similar situations exist with *Horizontal Wave* (confused with *Draw X* and *Draw Tick*), *Draw X* (confused with *Draw Circle*) and *Jogging* (confused with *Forward Kick*).

Table 5: Performance comparison between methods with and without the part-based clustering.

Method	Avg ± Std	Best	Worst
With part-based clustering	87.05 ± 3.75	95.56	74.39
Without part-based clustering	73.08 ± 4.34	83.03	61.97

Table 6: Performance comparison among different time scales for feature extraction. The notation $1, 5, 10$ means that three previous frames (1, 5, 10) before the current frame are used.

Time scale	Avg $\pm$ Std	Best	Worst
1	81.51 ± 3.66	89.59	70.38
$1, 5$	84.88 ± 3.35	92.67	75.61
$1, 5, 10$	87.05 ± 3.75	95.56	74.39
$1, 5, 10, 15$	82.31 ± 4.00	93.70	72.47
$1, 5, 10, 15, 20$	79.78 ± 4.28	93.70	68.77

4.3.2. Time Scale for Motion Feature Extraction

As described in Section 3.2, to take the time scale into consideration, we calculate the DP component of the motion features with several previous frames before the current frame. To determine how many of previous frames are sufficient, we repeat the first type of experiments on MSRAction3D with different numbers of previous frames for motion feature extraction and compare the recognition accuracy in Table 6. We can observe that taking the time scale into consideration could enhance representability of the motion features, which in turn results in higher recognition accuracy. Besides, using three previous frames (1, 5, 10) provides the best performance in this experiment. Take the action *Bend* as an example. Without taking the time scale into consideration, it only achieves 58.3% recognition rate. While using two previous frames (1, 5) already improves it up to 91.7% and using three previous frames (1, 5, 10) even manages to achieve 100% recognition rate. However, using more previous frames (15, 20) leads to worse performance,

which is mainly because the previous frames far away could not provide valuable information for recognition but produce more noise.

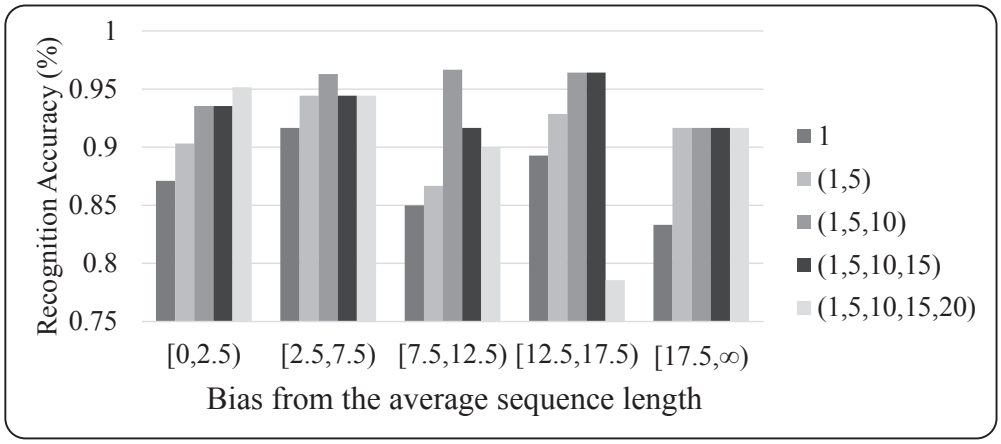


Figure 10: Recognition accuracies of various sequence lengths using different time scales for motion feature extraction. In the figure, "Bias from the average sequence length" means the absolute difference between the length of a sequence and the average length of all sequences of this action, according to which we divide the test set into several groups and compare the recognition accuracies for each group using different time scales.

In order to investigate how to decide the time scale for a given dataset, we record the sequence lengths of the samples of all 20 actions. We find out that for this dataset, the standard deviations of sequence lengths of most actions are around 10, which explains why using up to 10 previous frames is sufficient to represent the time scale. We further partition the test set into several groups of sequences according to the difference between the length of a sequence and the average sequence length, that is, according to the bias from the average sequence length. For each group, we then compare the

recognition accuracies obtained from using different time scales for motion feature extraction. The comparative results are plotted in Fig. 10 (accuracies are obtained from the best result). We can observe that extracting motion features for three previous frames (1, 5, 10) improves the accuracies for different sequence lengths. Moreover, the improvement is relatively prominent for sequences whose length biases from the average are around 10 ([7.5, 12.5)). These observations imply that, given a dataset, using the time scale up to the standard deviations of sequence lengths is highly possible to offer an optimal performance.

4.3.3. Action Graphs

The Viterbi decoding algorithm used for action graphs is a dynamic programming algorithm, which could output scores (confidences) for all actions at any frame. This suggests that we may make use of this benefit to recognize actions at earlier time before the end of an action with sufficient confidence or to automatically and reliably segment actions along with action recognition. This is also useful to make the recognition latency lower when we apply the recognition algorithm in real-time applications.

Here we carry out an experiment to verify the early detection performance of our method. In this experiment, we rescale the length of all test action sequences to be 1, and recognize the action at the points of 5/6, 2/3, 1/2, 5/12, 1/3, 1/4, respectively. (Here we do not set the points to make equal segments, just because the recognition rate usually does not vary equably. Thus we sample more points for the early stage where the recognition rate varies more rapidly.) The training procedure is just the same as before. We record the recognition rates at different points for all folds of subjects and

put the average results and the standard deviation together to draw Fig. 11. We can observe that, from right to left, at first the recognition rate goes down slowly with the recognition point moving earlier, and then it drops quickly when the point is earlier than 2/3. So with bearable decline of recognition accuracy (about 4%), we could achieve early detection of actions at the 2/3 point of the action sequence. At this point, the performance of our method is still higher than [9] (see Table 1).

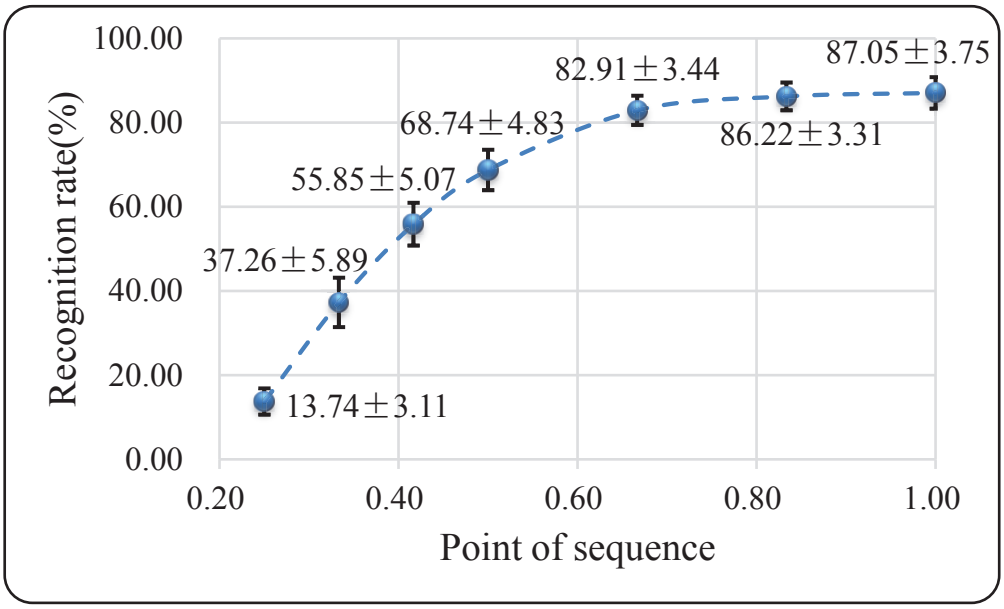


Figure 11: Recognition rates at different points of action sequences. We rescale the length of all test action sequences to be 1, and a point means the length used for prediction.

To summarize up the module analysis, the introduction of the part-based clustering in our proposal and modifying the motion features of [12] with consideration of time scale are demonstrated to be effective on the MSRAction3D

dataset, resulting in better performance. Meanwhile, we suggest a statistical principle for deciding the time scale for any specified dataset. Furthermore, early detection is proved to be feasible with the use of action graphs.

4.4. Impact of Parameters

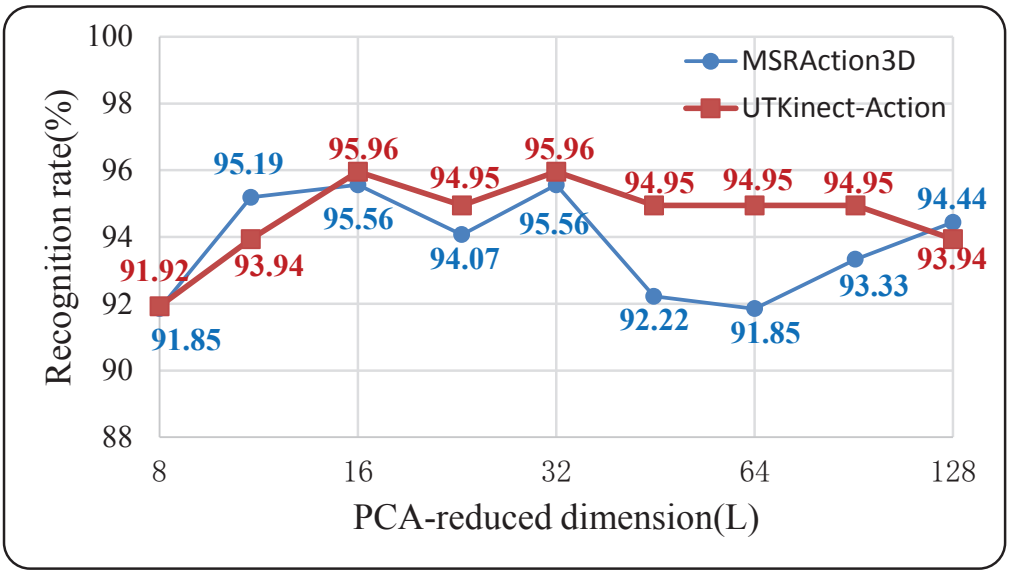
To verify the performance of our method versus different parameter values (here parameters include the PCA-reduced dimension L and the posture number K), we carry out experiments on MSRAction3D (the first type) and UTKinect-Action with various parameter values for the fold that gets the best result.

4.4.1. Impact of the PCA-reduced Dimension L

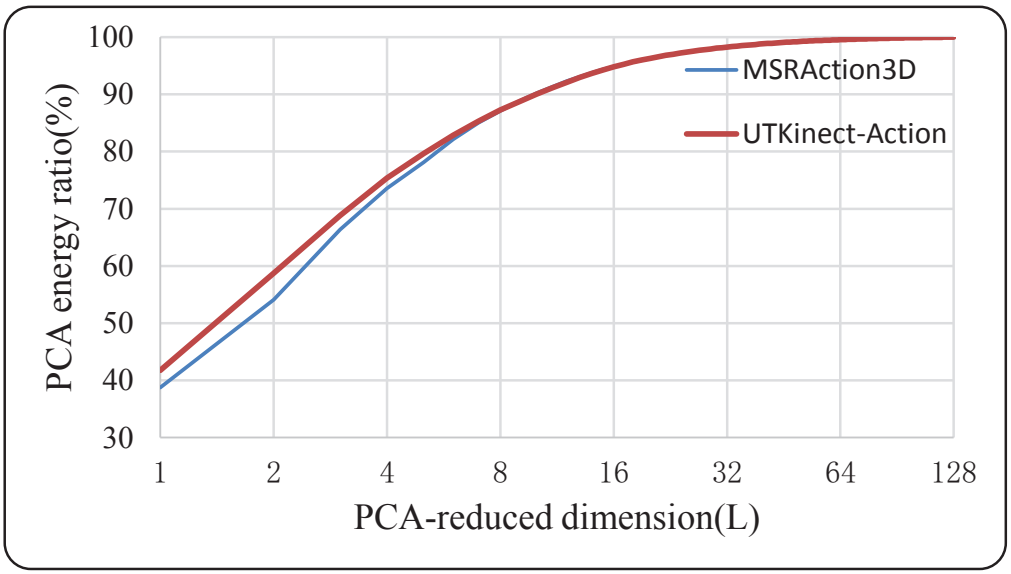
Fig. 12a shows how the performance of our method varies with different values of the PCA-reduced dimension, L. We can observe that, for both datasets, when L is small the performance slightly increases with L, and when L is large enough the performance becomes rather stable and then slightly decreases. This may be because when L is sufficiently large we can capture most of the energy as shown in the energy graph, Fig. 12b, and then more unimportant principal components induce certain irrelevant information. When L reaches 16, we already capture 95% of the energy and achieve the best performance. For this reason and for the consideration of small computational costs, we choose 16 as the value of L for both datasets.

4.4.2. Impact of the Posture Number K

Fig. 13 shows the impact of the number of postures, K, on the recognition performance. As we need more postures to characterize more actions, we take ratio $\rho = K/N$ as the parameter to be discussed, where N is the number of



(a) Impact of the PCA-reduced dimension on recognition performance.



(b) PCA energy graph. In the graph, "PCA energy ratio" is defined as $\frac{\sum_{i=1}^{L}\lambda_i}{\sum_i \lambda_i}$, where λ_i are the eigenvalues obtained from PCA and they are ordered as $\lambda_i \geq \lambda_{i+1}$.

Figure 12: Impact of the PCA-reduced dimension.

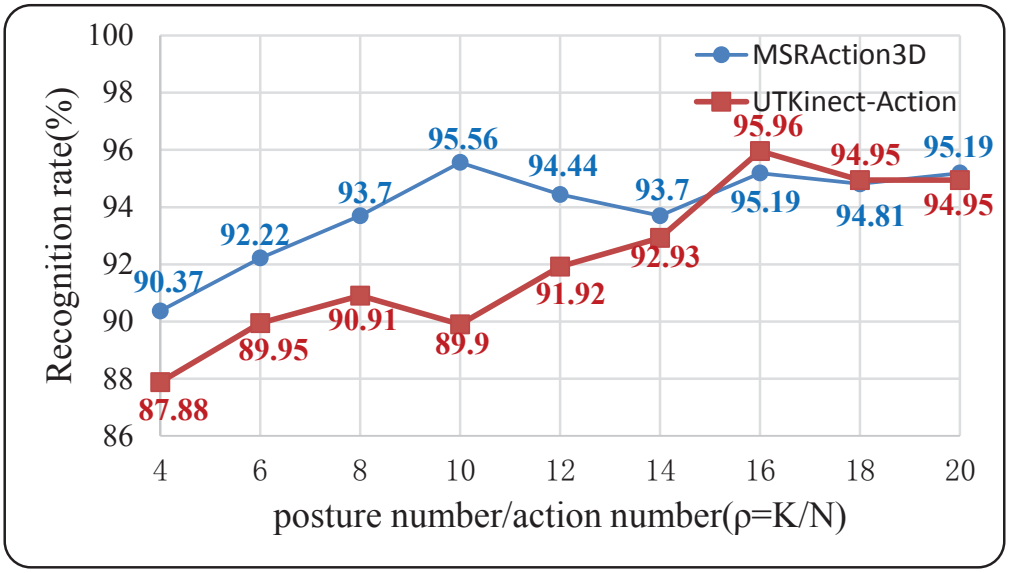


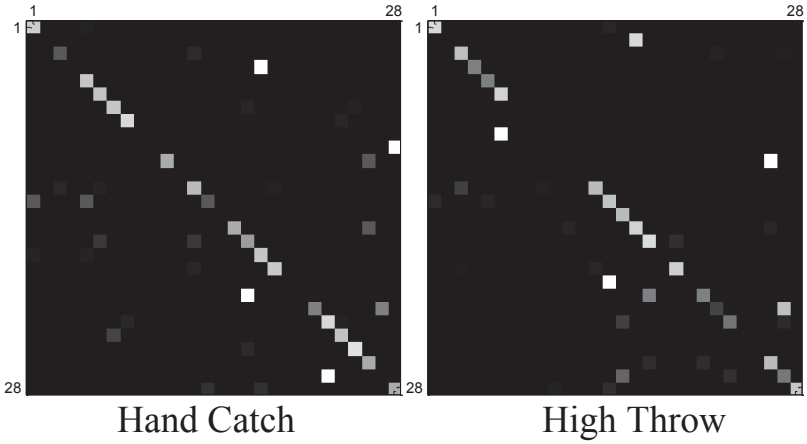
Figure 13: Impact of ρ on the performance. Here $\rho = K/N$, where N is the number of actions and K is the number of postures.

actions. As we can observe in Fig. 13, as ρ increases, the performance at first improves significantly and then stays at a high level.

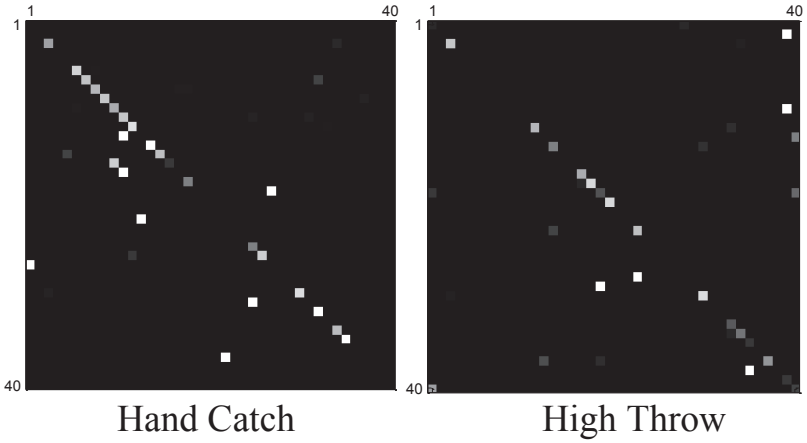
To explain this pattern, take two actions *Hand Catch* and *High Throw* in the MSRAction3D dataset as an example. When ρ is too small, we do not have enough postures to characterize all the actions so the transitional probability matrices of them are similar, which results in a poor recognition performance. Fig. 14a shows the main part of the transitional probability matrices of the two actions when ρ is too small (4). The two matrices seem similar to each other, which leads to a low recognition rate of 25% for *Hand Catch*. As ρ gets large enough, we manage to characterize different actions with different postures and/or different transitional probability matrices, which makes the recognition task much easier. Fig. 14b shows the main part of the transitional probability matrices of the same two actions when ρ is large enough (10). Compared with those in Fig. 14a where ρ is too small, the two matrices now are much more different from each other, which improves the recognition rate of *Hand Catch* to 75%. To verify the visual impression of the matrix similarity in Fig. 14a and Fig. 14b, we calculate the correlation r of the two transitional probability matrices, A and B, for these two actions, using the following formula:

$$r = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(A_{i,j}-\bar{A})(B_{i,j}-\bar{B})}{\sqrt{\left(\sum_{i=1}^{N}\sum_{j=1}^{N}(A_{i,j}-\bar{A})^2\right)\left(\sum_{i=1}^{N}\sum_{j=1}^{N}(B_{i,j}-\bar{B})^2\right)}},$$

where $\bar{A} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}A_{i,j}}{N^2}, \bar{B} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}B_{i,j}}{N^2}$. When ρ is 4, r is 0.44; when ρ is 10, r is 0.20, which indicates that the matrices are much less similar. Considering Fig. 13 and reasonable computational costs, we set ρ to 10 for



(a) When ρ is too small (4), matrices of two actions are moderately similar ($r = 0.44$).



(b) When ρ is large enough (10), two matrices are not similar ($r = 0.20$).

Figure 14: Transitional probability matrices of two actions at different values of ρ. The value at the (i, j) coordinate of the matrices represents the transitional probabilities from posture i to posture j. Each value is illustrated by the brightness, with a brighter one for a larger probability.

the MSRAction3D dataset and 16 for the UTKinect-Action dataset.

In summary, all the above experiments have verified that our proposed method has tackled to some extent the three challenges mentioned at the beginning of this paper. Firstly, with the utilization of the part-based clustering module, the challenge of high intra-class variance with low inter-class variance has been mostly solved. Secondly, we have notably worked out the challenge of variable movement speed by considering time scales for motion features. Thirdly, since our method is based on 3D skeleton, its computation costs are relatively low, making the method applicable in real time.

5. Conclusion

In this paper, we have proposed a novel two-level hierarchical framework for action recognition with 3D skeleton sequences. In the framework, we have introduced a new part-based five-dimensional feature vector to mine the most relevant body parts for each action sequence and to cluster action sequences, have investigated the time scale of dynamics to optimally modify established motion features, and have devised a score function for the action inference based on action graphs. Our experiments have also verified that, compared with other state-of-the-art methods, the proposed method could achieve higher accuracy on the complex MSRAction3D dataset.

Nevertheless, the performance of our method still considerably depends on the accuracy of the skeleton positions, even though we only utilize the relevant joints. Besides, the two datasets used here provide known beginning and end of the actions, which are not available in real-world interactions. Hence, further to our work is to investigate the open problem of segment-

ing actions automatically, reliably and quickly by using action graphs, as suggested in Section 4.3.3.

Acknowledgements

The authors are grateful to two reviewers and Mr Hengkai Guo for their constructive comments, in particular for their suggestions which have led to a large improvement of section 4.3, section 4.4 and the readability of this paper. Thanks to Mr Andrew Mpapalika for proofreading. This work was partially supported by NSFC61271390 and 2015AA016304.

References

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

[2] G. Wang, X. Yin, X. Pei, C. Shi, Depth estimation for speckle projection system using progressive reliable points growing matching, Applied Optics 52 (3) (2013) 516–524.

[3] X. Yin, G. Wang, C. Zhang, Q. Liao, Learning the missing values in depth maps, in: International Conference on Optical Instruments and Technology (OIT2013), International Society for Optics and Photonics, 2013, pp. 904508–904508.

[4] K. Liu, C. Zhou, S. Wei, S. Wang, X. Fan, J. Ma, Optimized stereo matching in binocular three-dimensional measurement system using structured light, Applied Optics 53 (26) (2014) 6083–6090.

[5] C. Shi, G. Wang, X. Yin, X. Pei, B. He, X. Lin, High-accuracy stereo matching based on adaptive ground control points, Image Processing, IEEE Transactions on 24 (4) (2015) 1412–1423.

[6] W. Li, Z. Zhang, Z. Liu, Action recognition based on a bag of 3D points, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, IEEE, 2010, pp. 9–14.

[7] J. Wang, Z. Liu, J. Chorowski, Z. Chen, Y. Wu, Robust 3D action recognition with random occupancy patterns, in: Computer Vision–ECCV 2012, Springer, 2012, pp. 872–885.

[8] X. Yang, C. Zhang, Y. Tian, Recognizing actions using depth motion maps-based histograms of oriented gradients, in: Proceedings of the 20th ACM International Conference on Multimedia, ACM, 2012, pp. 1057–1060.

[9] O. Oreifej, Z. Liu, HON4D: Histogram of oriented 4D normals for activity recognition from depth sequences, in: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE, 2013, pp. 716–723.

[10] H. Rahmani, A. Mahmood, D. Q Huynh, A. Mian, HOPC: Histogram of oriented principal components of 3D pointclouds for action recognition, in: Computer Vision–ECCV 2014, Springer, 2014, pp. 742–757.

[11] J. Wang, Z. Liu, Y. Wu, J. Yuan, Mining actionlet ensemble for action recognition with depth cameras, in: Computer Vision and Pattern

Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 1290–1297.

[12] X. Yang, Y. Tian, Eigenjoints-based action recognition using naive-Bayes-nearest-neighbor, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on, IEEE, 2012, pp. 14–19.

[13] L. Xia, C.-C. Chen, J. Aggarwal, View invariant human action recognition using histograms of 3D joints, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on, IEEE, 2012, pp. 20–27.

[14] C. Wang, Y. Wang, A. L. Yuille, An approach to pose-based action recognition, in: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE, 2013, pp. 915–922.

[15] M. A. Gowayyed, M. Torki, M. E. Hussein, M. El-Saban, Histogram of oriented displacements (HOD): describing trajectories of human joints for action recognition, in: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, AAAI Press, 2013, pp. 1351–1357.

[16] Y. Zhu, W. Chen, G. Guo, Fusing spatiotemporal features and joints for 3D action recognition, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on, IEEE, 2013, pp. 486–491.

[17] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, A. Del Bimbo, Space-time pose representation for 3D human action recognition, in: New Trends in Image Analysis and Processing–ICIAP 2013, Springer, 2013, pp. 456–464.

[18] H. Chen, G. Wang, L. He, Accurate and real-time human action recognition based on 3D skeleton, in: International Conference on Optical Instruments and Technology (OIT2013), International Society for Optics and Photonics, 2013, pp. 90451Q–90451Q.

[19] R. Vemulapalli, F. Arrate, R. Chellappa, Human action recognition by representing 3D skeletons as points in a Lie group, in: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE, 2014, pp. 588–595.

[20] M. Barnachon, S. Bouakaz, B. Boufama, E. Guillou, Ongoing human action recognition with motion capture, Pattern Recognition 47 (1) (2014) 238–247.

[21] I. Kapsouras, N. Nikolaidis, Action recognition on motion capture data using a dynemes and forward differences representation, Journal of Visual Communication and Image Representation 25 (6) (2014) 1432–1445.

[22] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, R. Bajcsy, Sequence of the most informative joints (SMIJ): A new representation for human skeletal action recognition, Journal of Visual Communication and Image Representation 25 (1) (2014) 24–38.

[23] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, A. Blake, Efficient human pose estimation from single depth images, Pattern Analysis and Machine Intelligence, IEEE Transactions on 35 (12) (2013) 2821–2840.

[24] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moore, Real-time human pose recognition in parts from single depth images, Communications of the ACM 56 (1) (2013) 116–124.

[25] L. He, G. Wang, Q. Liao, J.-H. Xue, Depth-images-based pose estimation using regression forests and graphical models, Neurocomputing 164 (2015) 210–219.

[26] W. Li, Z. Zhang, Z. Liu, Expandable data-driven graphical modeling of human actions based on salient postures, Circuits and Systems for Video Technology, IEEE Transactions on 18 (11) (2008) 1499–1510.