# 16811 Project Writeup: 3D Trajectory Representation for Understanding Human Actions

Kenan Deng (kenand), Chunhui Liu (chunhui2)

12/14/2018

## 1 Introduction

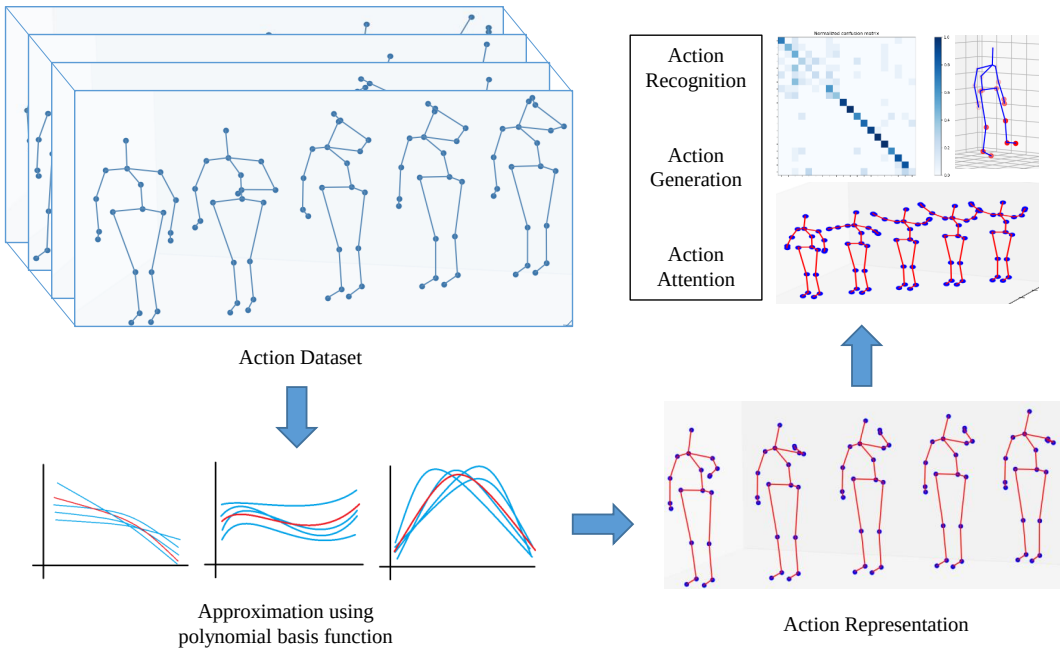


Figure 1: **Overview**, learning action3D skeleton points trajectories use approximation approaches with polynomial basis functions for action understanding.

Nowadays, as the development of cost-efficient depth camera [1] and real-time pose estimation algorithm [10] progresses, 3D skeleton points have been widely available. 3D skeleton points derived from the human body can provide valuable and comprehensive representations of human actions [3]. Modeling the trajectories of real-world skeleton points can help us understanding and generating human actions.

In this project, we focus on representing trajectories of 3D points efficiently using approximation approaches. Given real-world action coordinates, we try to represent the action trajectories using a combination of basis functions. There are following questions that will be possibly concerned: How to generate new trajectories on new animation models given real-world data? Given a human skeleton data, is there a way to classify those action? How important is each joint in the animation and how to assign attention weight to each joint?

Illustrated as in 1, and inspired by approximation technique used in class, our approach focus on deriving a parameterization scheme of the 3D skeleton points trajectories. We proposed 3 schemes: parameterization by absolute joints positions, parameterization by body parts with relative axis-angle representations, and parameterization by temporal joints position difference. In all 3 schemes, we approximate the trajectory of the parameters using a set of polynomial function basis. Finally, we experiment these parametreization schemes along with approximation result on data set and test their performance. [1]

# 2 Related Work

Action understanding in videos is intensively studied area both in academia and industry, with broad applications in human-machine interaction, video surveillance and robotics. A key category of action understanding methods aims at analyzing human actions in RGB videos [11, 13]. In the recent years, with the advent of affordable color-depth sensing cameras, such as RealSense and Kinect, there is an increasing amount of visual data containing multi-feature(*e.g.*, RGB, depth, IR, skeletons).

Different feature contain modal-specific characteristics and can be utilized to adapt different application scenarios. For skeleton data, according to the biological research [4], it can provide the valuable and comprehensive representation of a series of human dynamics. Due to its unique characteristics, it shows great robustness to illumination variation, clustered background, and camera motion. Because skeleton data is of lower dimension compared with RGB features, to model its temporal dynamics can be much more time-efficient and power-efficient. The low dimension of 3D skeleton data makes it possible to achieve real-time computing. Several applications have been proposed for detecting and forecasting actions on the fly [7, 8].

By considering co-occurrences of local features [9, 2] and handcraft temporal representations [12, 15], semantic features for further classification can be extracted from skeleton representations. Some recent works have proposed new low-level graph based representations for classification [5, 14]. And in deep learning method, due to succinctness of skeleton data, recurrent neural network (RNN) models can be naturally used to model temporal dynamics. Applying RNN derived networks like

[1] We released our code on Github.

Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU) is one way to achieve better performance for skeleton based action recognition.

In this project, our motivation is not to simply recognize action category, but generally re-represent skeleton action using simple basis functions. Ideally, This feature is useful for general action understanding tasks, including action recognition, action compression, action generation and action attention.

# 3 Skeleton Data

3D skeleton data usually consists of joints positions in a coordinate frame over a span of time. In this section we review the data format of our main dataset used in this project: MSR Action Recognition Datasets[6] , as well as how to preprocess the data to normalize the input.

## 3.1 Data format

The MSR dataset consists of 20 actions. There are total 10 actors to perform these actions and for each actor, it performs the same action 3 times. For single action performed by an actors, the data is a time sequence of absolute joints position: for each joint $j$, a pair of world coordinates at time $t$ are recorded $\{x_t^j, y_t^j, z_t^j\}$ and are stored sequentially in a text file.

## 3.2 Preprocessing

To preprocess the data, first we move every skeleton into a common frame by fixing the neck joint position to the origin of the absolute world coordinate. This effectively remove the translation difference of different actors for the same animation. Then we select one actor as reference actor and normalize the bone length of all other skeleton to be the same as the reference skeleton. This should remove the difference of joints position caused by actors having different skeleton. Finally, for all data with the same action, there can be some temporal misalignment. We remove this by extending all animations to 100 frames and the missing frames in the middle are filled using piece-wise linear interpolation.

# 4 Proposed Methods

## 4.1 Naive method

For a single animation of one action, given the set of data points for each joints $j \in J$ at time $t \in [0, T]$ : $\{x_t^j, y_t^j, z_t^j\}$, using a naive approach, we can express each coordinate as a function of time $t$: $x^j(t), y^j(t), z^j(t)$. This naive method models each

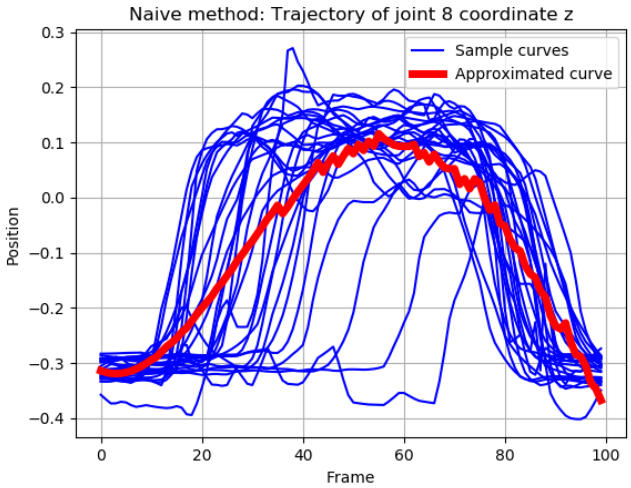


Figure 2: Using naive method to approximate sample trajectories

joint separately and each coordinate separately. Now the problem breaks down to $3|J|$ 1-D approximation problems.

For each 1-D approximation problem, suppose the function we are trying to approximate is $f^j(t)$, where $f^j(t)$ can be any $x, y, z$ coordinate, and supposed we have $N$ samples for this action, we can formulate a least square approximation scheme using a set of $M$ basis function $b_0(t), b_1(t), ...b_m(t)$ as follows:
For a single example:

$$f^j(t) \approx c_0 b_0(t) + c_1 b_1(t) + ...c_m b_m(t)$$

$$\begin{bmatrix} b_0(t_0) & b_0(t_1) & ... & b_0(T) \\ b_1(t_0) & b_1(t_1) & ... & b_1(T) \\ ... & & & \\ b_m(t_0) & b_m(t_1) & ... & b_m(T) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ ... \\ c_m \end{bmatrix} = \begin{bmatrix} f^j(t_0) \\ f^j(t_1) \\ ... \\ f^j(T) \end{bmatrix}$$

For N examples, we simple repeat the left matrix by N times and stack them vertically, also we stack all samples on the right hand side vertically, now the linear system becomes:

$$\begin{bmatrix} b_0(t_0) & b_0(t_1) & ... & b_0(T) \\ ... & & & \\ b_m(t_0) & b_m(t_1) & ... & b_m(T) \\ b_0(t_0) & b_0(t_1) & ... & b_0(T) \\ ... & & & \\ b_m(t_0) & b_m(t_1) & ... & b_m(T) \\ ... & & & \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ ... \\ c_m \end{bmatrix} = \begin{bmatrix} f_0^j(t_0) \\ f_0^j(t_1) \\ ... \\ f_0^j(T) \\ ... \\ f_N^j(t_0) \\ f_N^j(t_1) \\ ... \\ f_N^j(T) \end{bmatrix}$$

To solve this linear system, provided that the rank of left matrix is greater than number of basis function used $M$, can be solved either using SVD or pseudo inverse. And this should gives us a least square solution to the system. Apply the same procedure for each coordinate and joint, we now have a list of coefficients of $3M|J|$ that correspond to the basis function for each coordinates and joint. We can easily reconstruct the representative curve by calculating the product of basis function and its coefficient. In figure 4 we show a approximation result of one joint.

## 4.2 Axis angle method

The axis angle method is inspired by the rotation representation of two vectors. Any two vectors are related by an axis of rotation and a rotation angle. See figure 3

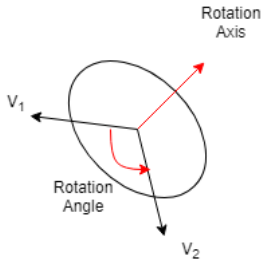


Figure 3: The axis-angle representation of rotation

For a skeleton with J joints $Joints_0, ...Joints_J$, first we define an tree ordering of the joints. Now each bone inside the skeleton can be represented use this tree ordering as a pair of a node and its parent $(parent(Joints_i), Joints_i)$. Two bones $i, j$ are connected if first node of the bone is the children of the other bone's second node: $Joints_j = parent(Joints_i)$. Now we can convert the skeleton data into a set of rotation axis and rotation angles by the followings algorithm 1

**Algorithm 1** Calculating axis angle

1: Initialize the tree structure of the bone
2: Store all bone length
3: Fix the root bone location as reference
4: **repeat**
5: Get a connected bone pair $P : (J_i, J_j), Q : (J_j, J_k)$
6: Calculate the vector as $v_p = J_i - J_j$, $v_q = J_k - J_j$
7: Calculate the rotation axis using cross product $a = v_p \times v_q$
8: Calculate the rotation angle using dot product $angle = \arccos(\frac{v_p \cdot v_q}{|v_p||v_q|})$
9: store in dictionary the axis $a$, angle $angle$
10: **until** All bone pair are calculated

This is a body part based method. One advantage of body part method over joint method is that the physical constraints of the bones are automatically encoded.

The result of this method are set of angle and rotation axis vectors for bone pair $P, Q$, angle: $\theta_{P,Q}$ axis:$\vec{a}_{P,Q} = (a_x, a_y, a_z)$, To approximate the angle change and axis change over time, we derive the same approach as naive method: Using a set of basis function to approximate $\{a_x(t), a_y(t), a_z(t), \theta_{P,Q}(t)\}$ separately.

## 4.3 Temporal difference method

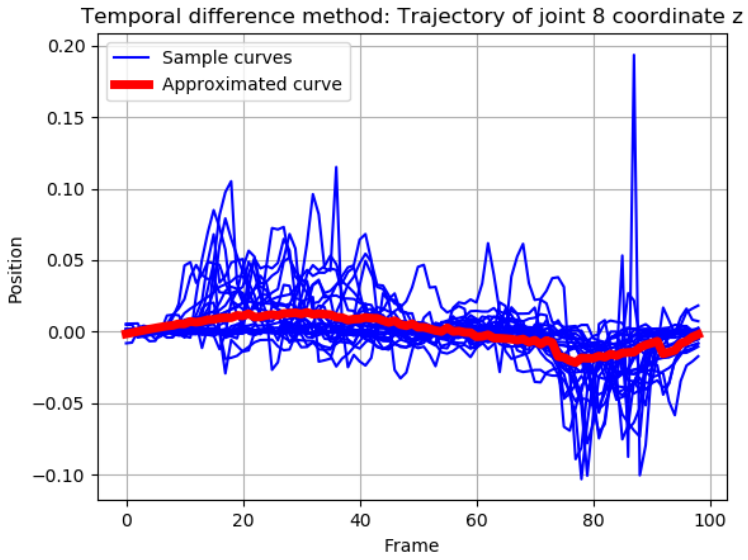


Figure 4: Using temporal difference method to approximate sample trajectories

As mentioned in section 2, many action recognition approaches focus on modeling temporal change using local feature or deep networks. Recall the process we understand actions, the change of locations may be more important than the exact locations. Thus, we introduce temporal difference method for our approximation method. The key idea is not approximate the location using basis functions, but the change of locations. This can greatly avoid location variance for same action performed by different subjects.

The temporal difference method is an extension to the naive method. The main difference is that we parameterize the trajectory $f^j(t)$ by its temporal difference: $df^j(t)$ and keep the $f^j(t_0)$ in order to reconstruct the skeleton data. Here, we use the forward difference $df^j(t) \approx f^j(t_{i+1}) - f^j(t_i)$ to approximate the temporal derivative. Then, we apply the same least square principle from naive method and use a set of basis function to approximate the forward difference.

# 5 Experimental Results

In this section, we briefly introduce three application using our representation method. We test our method on three tasks: action generation, action recognition and action attention.

We use MSR action 3D Daset [6] for following experiments. It has 20 action categories, Each action was performed by ten subjects for three times. The subjects were facing the camera during the performance. Altogether, the dataset has 565 action samples.

## 5.1 Action Generation

In this section, we test our action representation method for generation ability. Basically, this result illustrates how our method learn a overall representation of samples with same label in a dataset by integrating shared information and ignoring noise.

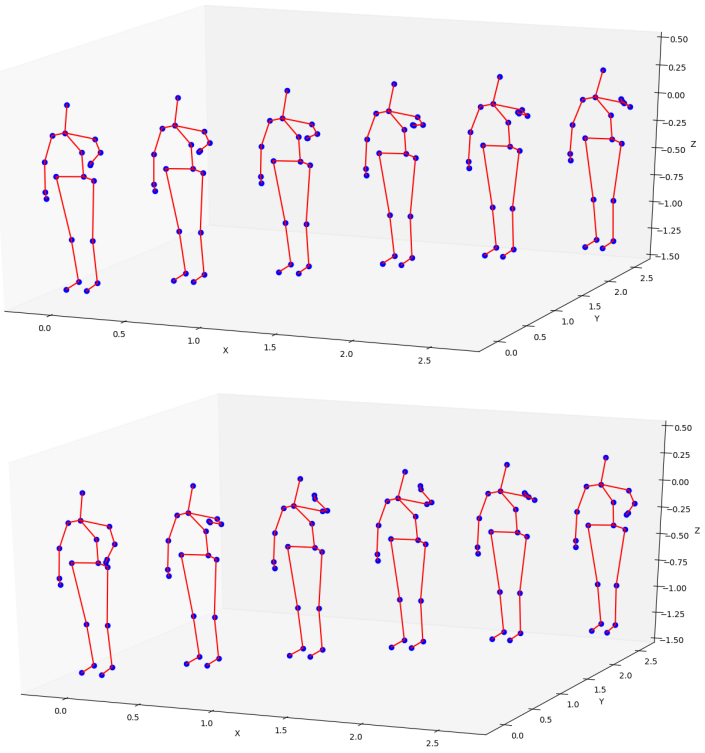


Figure 5: Action generation result (high arm wave). top: action generation result using naive method; bottom: action generation result using temporal difference learning. We can observe that the joints don't change much in top result, due to the noise in dataset. However, temporal difference learning perform well.

We include our qualitative result in figure 5 and on Youtube Link. In youtube video, we can observe that without introducing normalization, the generated results will shaking too much. And using naive method, the change of joints position is small due to the noise in data.

## 5.2 Action Recognition

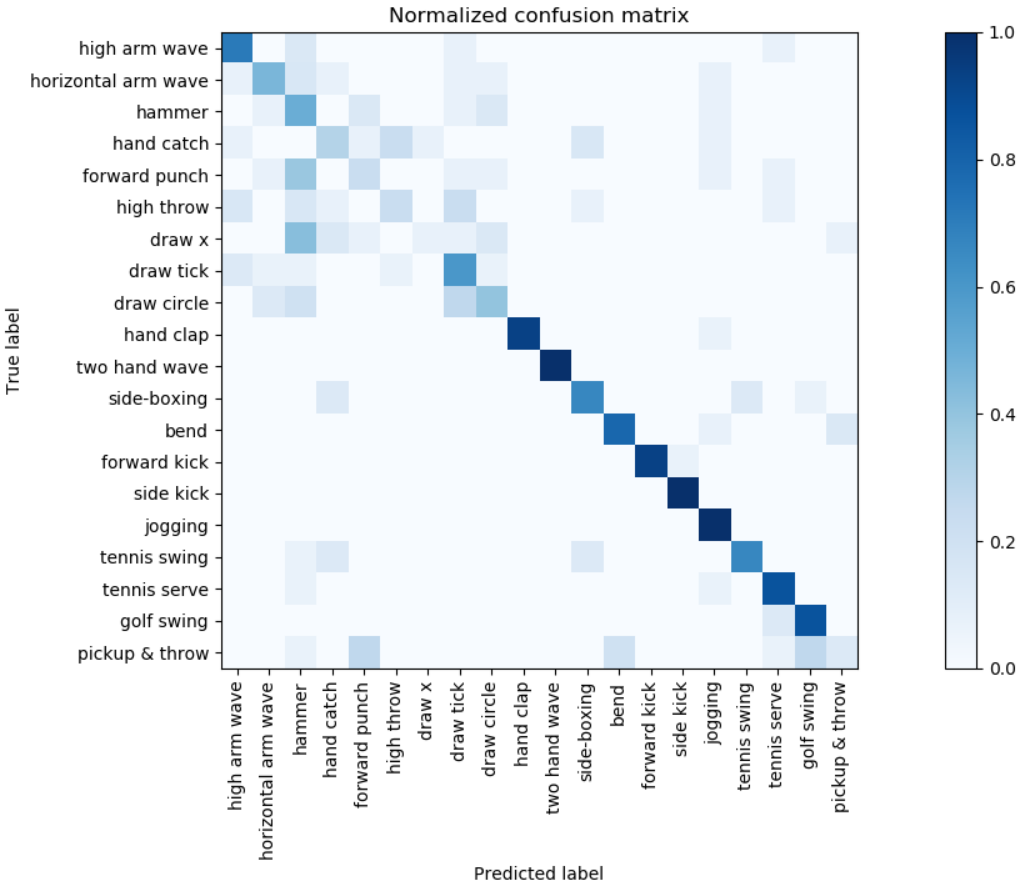


Figure 6: Confusion Matrix, we can observe that first several categories only focus on the action of hands, leading to bigger error. The following actions include the dynamics of other human part, leading to better classification result. In section 5.3, we will further show that first 9 action categories share almost same action attention.

In this section, we test our methods on classification task, namely action recognition. We compute 20 high-level representation as training set for nearest neighbor approach. This is super fast for testing with $O(M)$ time complexity where $M$ is the number of labels. To evaluate different approaches, we utilize top-1 error and top-3

| Method | Top-1 Accuracy | Top-3 Accuracy |
|---|---|---|
| Naive Method | 43.82% | 65.02% |
| Naive Method + Normalization | 47.00% | 72.79% |
| Temporal Difference | 55.48% | 75.62% |
| Temporal Difference + Normalization | 62.19% | 81.98% |

Table 1: Action classification Result

error for classification. We split the MSR Action 3D dataset to 280 training samples and 285 testing samples.

We include a confusion matrix in figure 6 to show the performance of our final method. Moreover, to show the importance of different part in our approach, we include ablation results in table 1.

## 5.3 Action Attention

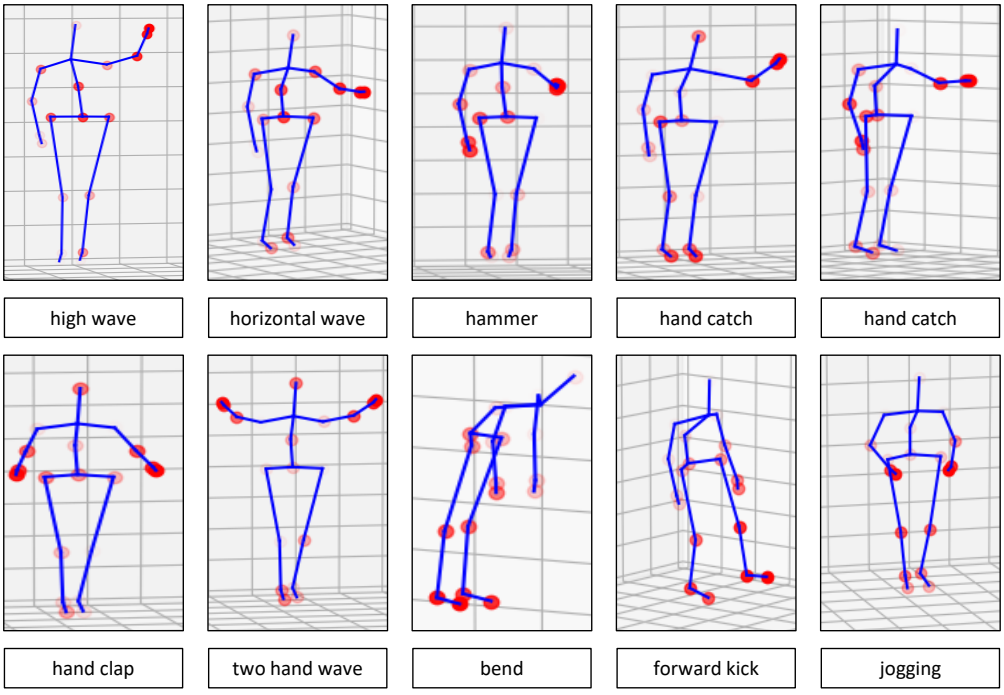


Figure 7: Action attention results.

Action attention is a relative new field in action understanding area. The idea is that to understand a action trajectory, we need pay different attention to different part. This is easy to be accomplished using our representation method, as our method only compress the dimension of sample numbers and keep all feature space.

Our coordinate of different basis function imply the moving intensity of each joint. We include our attention results in figure 7.

# 6 Conclusion and Discussion

In this project, we explore action understanding task using approximation method introduced in the class. We learn a trajectory representation using a set of training data. We introduce three methods to better capture action dynamics. We test our presentation approach for three tasks, action generation, action recognition, and action attention. We include qualitative and quantitative results to show the effectiveness of our method.

This method can also be interpreted as a compressing method. Different from other methods like PCA and Auto-encoder, which compress feature dimension, our method focuses on compress the dimension of data samples. This give us the opportunity to use this method as general feature extractor without losing any information of one sample. The drawback of this approach is that this method will fail if the variance of samples in one class is big.

# References


[1] Microsoft corp. redmond wa. kinect for xbox 360.

[2] Yusuke Goutsu, Wataru Takano, and Yoshihiko Nakamura. Motion recognition employing multiple kernel learning of fisher vectors using local skeleton features. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 79–86, 2015.

[3] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics*, 14(2):201–211, 1973.

[4] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics*, 14(2):201–211, 1973.

[5] Piotr Koniusz, Anoop Cherian, and Fatih Porikli. Tensor representations via kernel linearization for action recognition from 3d skeletons. In *European Conference on Computer Vision*, pages 37–53. Springer, 2016.

[6] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 9–14. IEEE, 2010.

[7] Yanghao Li, Cuiling Lan, Junliang Xing, Wenjun Zeng, Chunfeng Yuan, and Jiaying Liu. Online human action detection using joint classification-regression recurrent neural networks. In *European Conference on Computer Vision*, pages 203–220. Springer, 2016.

[8] Chunhui Liu, Yanghao Li, Yueyu Hu, and Jiaying Liu. Online action detection and forecast via multitask deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 1702–1706. IEEE, 2017.

[9] Lorenzo Seidenari, Vincenzo Varano, Stefano Berretti, Alberto Bimbo, and Pietro Pala. Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 479–485, 2013.

[10] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1297–1304. Ieee, 2011.

[11] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. 2014.

[12] Lingling Tao and René Vidal. Moving poselets: A discriminative and interpretable skeletal motion representation for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 61–69, 2015.

[13] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: towards good practices for deep action recognition. 2016.

[14] Pei Wang, Chunfeng Yuan, Weiming Hu, Bing Li, and Yanning Zhang. Graph based skeleton motion representation and similarity measurement for action recognition. In *European Conference on Computer Vision*, pages 370–385. Springer, 2016.

[15] Mihai Zanfir, Marius Leordeanu, and Cristian Sminchisescu. The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2752–2759, 2013.