Proceedings of the 2018 13th World
Congress on Intelligent Control and Automation
July 4-8, 2018, Changsha, China

Kinematics Features for 3D Action Recognition Using Two-Stream CNN

Jiangliu Wang and Yunhui Liu

Abstract— **Due to the great success of convolutional neural networks (CNN) on image classification problems, several attempts have been made to train deep neural networks for human action recognition problem. But since CNN is designed for static RGB images, it is not easy for it to learn temporal information from videos. To tackle this problem, temporal encoded kinematics features are proposed, which compute the linear velocity and orientation displacement based on human skeleton data. A two-stream CNN architecture is used, incorporating spatial and temporal networks. The spatial ConvNet is trained on still RGB images, while the temporal ConvNet is trained on the proposed encoded kinematics features. We evaluate our method on a popular and challenging 3D multi-view human action benchmark, Northwestern-UCLA dataset. The experiment results show that our proposed method is fast to train and performs better when compared to traditional handcrafted features.**

I. INTRODUCTION

Human action recognition is one of the most challenging problems in the field of computer vision and has attracted lots of attention in recent decades [1]–[3]. The applications of this field of research can vary from elderly caring at home to large scale surveillance monitoring in public places.

Methods proposed to solve action recognition problem can be roughly classified into two categories: handcrafted feature method and deep learning method. The handcrafted feature method extracts local features that can represent the spatial and temporal information of human action and then trains discriminative classifiers, such as Support Vector Machines (SVM), to finally predict the label of different actions. Laptev first proposed the spatial-temporal interest points [1] method, which is one the most remarkable methods in the early ages of investigating human action recognition. Following the design concept of this method, more and more effective detectors are developed [4]–[6]. They are proposed to overcome the challenges in action recognition, including human scale variance, light condition change, etc. Apart from the space-time interest points method, Wang et al. [7] propose a dense-trajectory based method. This method samples dense points from each frame and then tracks them based on a dense optical flow filed. A descriptor based on motion boundary histograms is finally designed to encode the trajectory information. And the authors further improved this work by considering the camera motion [8]. SURF descriptors and dense optical flow are used to match feature points between frames to estimate camera motion. And finally the camera motion is canceled out from the trajectories. The dense-trajectory based method is proved to be the most successful handcrafted method and achieves the state-of-art recognition result among all the handcrafted methods.

Deep learning has achieved impressive results on image classification [9] and object detection [10]. Several attempts have been made to apply deep learning for action recognition. The deep learning method is an end-to-end method, which automatically extracts action representation using deep neural networks. There are mainly two popular neural network architectures: 3D CNNs [11]–[13] and two-stream CNNs [2], [14], [15]. Inspired by the traditional 2D CNNs for single image classification, 3D CNNs extend it to 3D convolution layers and 3D pooling layers to extract spatial-temporal representation. But this network architecture itself is usually hard to train and thus too shallow to learn powerful representations. Two-stream CNNs do better when concerning performance on public benchmarks. The idea of this method comes from two-stream hypothesis proposed by neurologists [2]. It is reported that human visual system consists of two pathways: the ventral stream to recognize still object and the dorsal stream to recognize motion. To imitate such a visual system, the two-stream CNN method first trains two convolution neural networks on RGB images and stacked optical flow, respectively and then fuses these two networks to get the final classification results. One drawback of this method is that the computation of the optical flow is quite time-consuming.

Due to the advent of low-cost and real-time depth sensors, such as kinect, the problem of human pose estimation becomes much more easier. Shotton proposed a real time pose estimation method that can predict the 3D coordinates of human skeletons accurately from one single depth image [16]. And the method is adopted in the kinect sensor. Since then, action recognition research based on skeleton data has attracted more and more interests [17]–[19]. Traditional powerful CNN method is not suitable for this problem since it requires image as input while the skeleton data is only sequence of numbers. To solve this problem, recurrent neural network(RNN) is adopted to learn action representation from skeleton data, which is originally designed for natural language processing. Compared to RGB image, skeleton data is more robust to view, appearance and light condition variation. And it is fast to process since human pose can be described by 20 or 25 joints. But when concerning human-object interaction, only skeleton data itself is not sufficient to distinguish different actions. And also the RNNs method

All the authors are with CUHK T Stone Robotics Institute and the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, HKSAR. Corresponding author e-mail: yhliu@mae.cuhk.edu.hk

*This work is supported by the Hong Kong Research Grants Council and CUHK T Stone Robotics Institute.

978-1-5386-7346-1/18/$31.00 © 2018 IEEE

emphasizes too much on the temporal information while pays little attention to the spacial information, which could also lead to worse classification performance.

In this paper, we tackle the action recognition problem by capturing the complementary information on appearance from RGB data and motion from skeleton data using a two-stream CNN. Encoded kinematics features, which compute linear velocity and orientation displacement of human joints, are proposed to represent the temporal information based on the original skeleton data. These encoded kinematics features make it possible to train a temporal CNN and finally be fused with a spatial CNN. Our proposed method is validated on the Northwestern-UCLA dataset, which is a challenging multi-view RGB-D human action dataset [23]. Experiment results show that although these features are quite simple, they are fast to train and could achieve better performance than many handcrafted features.

The main contribution of this paper lies in three aspects: (1) kinematics features, including linear velocity and orientation displacement, are proposed to represent human action effectively; (2) an image encoding method is designed to make use of the power of CNN for action recognition based on skeleton data; (3) a key frame selection scheme is proposed to fit actions with different frame length into same image size for training.

The organization of this paper is described as follows. In section 2, we present how to generate the encoded kinematics features, including kinematics features extraction and image encoding. The neural network architecture is described in section 3. To evaluate our proposed method, a variety of experimental results are presented in section 4. Finally, we summarize our work in section 5.

II. ENCODED KINEMATICS FEATURES

CNN expects images as inputs. However, human skeleton data only provides 3D coordinates of each joint, which could not be fed into CNN directly. To solve this problem, we propose a novel method that can encode joint positions into 3-channel color images. The proposed method consists of two steps: (1): extract kinematics features from input skeleton sequences; (2): encode kinematics features into RGB images.

A. Kinematics Features Extraction

We adopt a typical human skeleton model with 20 joints, which can be tracked by kinect sensor v1. Similar to optical flow of the original two stream CNN, positions displacement, i.e., linear velocity and orientation displacement are investigated based on the 20 skeleton joints.

For an action video with N frames, at frame t, the 3D coordinates of the joint i can be obtained from human pose estimation [16]:

$$\boldsymbol{P}_t^i = (p_x, p_y, p_z), \quad i = [1;20], t = [1;N]. \tag{1}$$

The linear velocity of joint i at frame t is computed between itself and its following frame $t+1$:

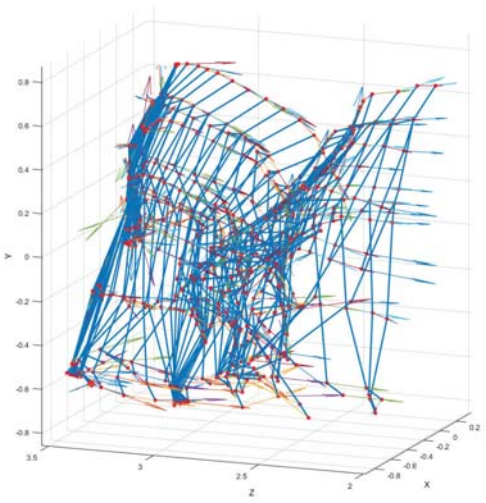

Fig. 1. Visualization of linear velocity computed from action *pick up* in the Northwestern-UCLA dataset.

$$\boldsymbol{v}_t^i = \boldsymbol{P}_{t+1}^i - \boldsymbol{P}_t^i, \quad i = [1;20], t = [1;N-1]. \tag{2}$$

To visualize the linear velocity more intuitively, the starting point of vector $\boldsymbol{v}_t^i$ is put at $\boldsymbol{P}_t^i$, as shown in Fig 1.

The orientation displacement capture the angle variation of each joint. Given the 3D coordinates $\boldsymbol{P}_t^i$ of joint i at frame t , the corresponding orientation can be computed as:

$$\boldsymbol{\Phi}_t^i = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} arccos \frac{p_x}{\sqrt{p_x^2+p_y^2+p_z^2}} \\ arccos \frac{p_y}{\sqrt{p_x^2+p_y^2+p_z^2}} \\ arccos \frac{p_z}{\sqrt{p_x^2+p_y^2+p_z^2}} \end{bmatrix}, \quad i = [1;20], t = [1;N]. \tag{3}$$

The orientation displacement of joint i at frame t is computed between itself and its following frame $t+1$:

$$\boldsymbol{\omega}_t^i = \boldsymbol{\Phi}_{t+1}^i - \boldsymbol{\Phi}_t^i, \quad i = [1;20], t = [1;N-1]. \tag{4}$$

The visualization of orientation displacement is shown in Fig. 2

Note that a feature with more physical meaning is angular velocity of each body part, which can be illustrated as follows:

$$v_1 = v_2 + \boldsymbol{\omega} \times r. \tag{5}$$

But three points are needed to compute the final angular velocity, which could not be achieved given only the positions of two joints.

B. Image Encoding

To feed the kinematics features into CNN, we need to encode these feature numbers into RGB images. Given an action video with N frames, after kinematics feature extraction, linear velocity $\boldsymbol{V}$ and orientation displacement $\boldsymbol{\Phi}$ are computed into $(N-1)\times 20\times 3$ matrix respectively, which just well suits the three channels RGB images with height $N-1$ and width 20.

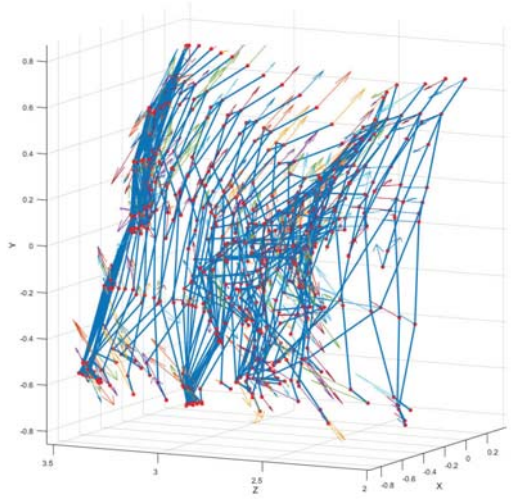

Fig. 2. Visualization of orientation displacement computed from action *pick up* in the Northwestern-UCLA dataset.

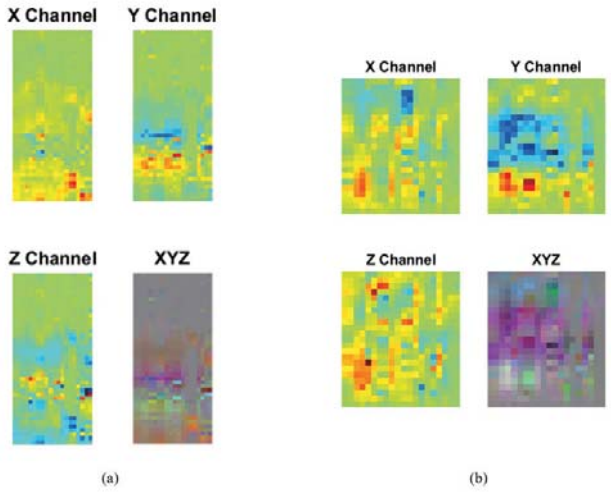

Fig. 3. Encoded linear velocity of two actions in the Northwestern-UCLA dataset. (a) *pick up* action; (b) *sit down* action.

Normalize all feature values to lie between 0 and 255 using:

$$X_{norm} = \frac{X - min(X)}{max(X) - min(X)} \times 255 \tag{6}$$

where $min(X)$ computes the minimum value of linear velocity or orientation displacement and $max(X)$ computes the maximum. Fig. 3 shows the encoded linear velocity of action *pick up* and action *sit down* from Northwestern-UCLA dataset. Note that the images are not presented in a low-resolution, but just the encoding output. The x, y, z channel images show the linear velocity in x, y, z axis, respectively. We use pseudo-color to show it in a more clear way, but in fact, they are gray image since all of the values lie between 0 and 255.

III. NETWORK ARCHITECTURE

The scheme of the proposed two stream CNN based on RGB and skeleton data is shown as Fig 4. It follows the basic framework of the original two stream CNN, with one CNN trained on RGB data and the other trained on encoded kinematics features. Each stream is implemented using a deep ConvNet and their softmax scores are combined by late fusion. Spatial stream ConvNet operates on individual RGB frames, whilst temporal stream ConvNet operates on encoded kinematics features computed from skeleton data. A VGG-16 like CNN is used as the layer configuration for both spatial and temporal ConvNet.

A. Spatial ConvNet

When training the still RGB images, we use a pre-trained model VGG16, which is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford [2]. The model is one of the most popular and successful network architecture on image classification problem, achieving 92.7% top-5 test accuracy in ImageNet [9] challenge.

VGG 16 expects 224 x 224 pixel RGB images as input and subtracts the mean image values, calculated over the entire ImageNet training set, from each image. It is a very deep network with a lot of convolution layer followed by max-pooling, reducing the dimensionality. The key feature of VGG16 is that each convolution layer performs 3x3 convolutions with stride 1 and pad 1, after which the image dimensionality remains the same. And each max-pooling layer performs with stride 2, pad 0, which reduces the image size in half.

When using VGG 16 to train RGB-D action datasets, we don't train from scratch since the data size is quite small, and may lead to over-fitting problem. Instead, we fine-tune the pretrained model on ILSVRC-2012. By fine-tuning, we mean that replace and retrain the classifier on top of the ConvNet on the new dataset, and fine-tune the weights of the pretrained network by continuing the backpropagation. And since in the last three fully convolutional layers, parameters are $7 \times 7 \times 512 \times 4096 = 102,760,448$ (fc1), $4096 \times 4096 = 16,777,216$ (fc2), and $4096 \times 1000 = 4,096,000$ (fc3), it is much easy to be over-fitting. To address this problem, we add drop-out layer [20] after each fully convolutional layer.

B. Temporal ConvNet

As illustrated before, the original VGG 16 ConvNet expects 224 x 224 pixel color images. However, for our encoded kinematics features, image size is $(N-1) \times 20 \times 3$, which is much smaller than the expected input size. Simple resize operation on encoded kinematics features could distort the original images and add more noise.

Inspired by another popular image classification dataset-CIFAR-10 dataset [21], which consists of 60000 32x32 tiny color images in 10 classes, we found that the VGG 16 network also works well and achieves 92.4% top-1 test accuracy [22]. Therefore, image transformation of the original

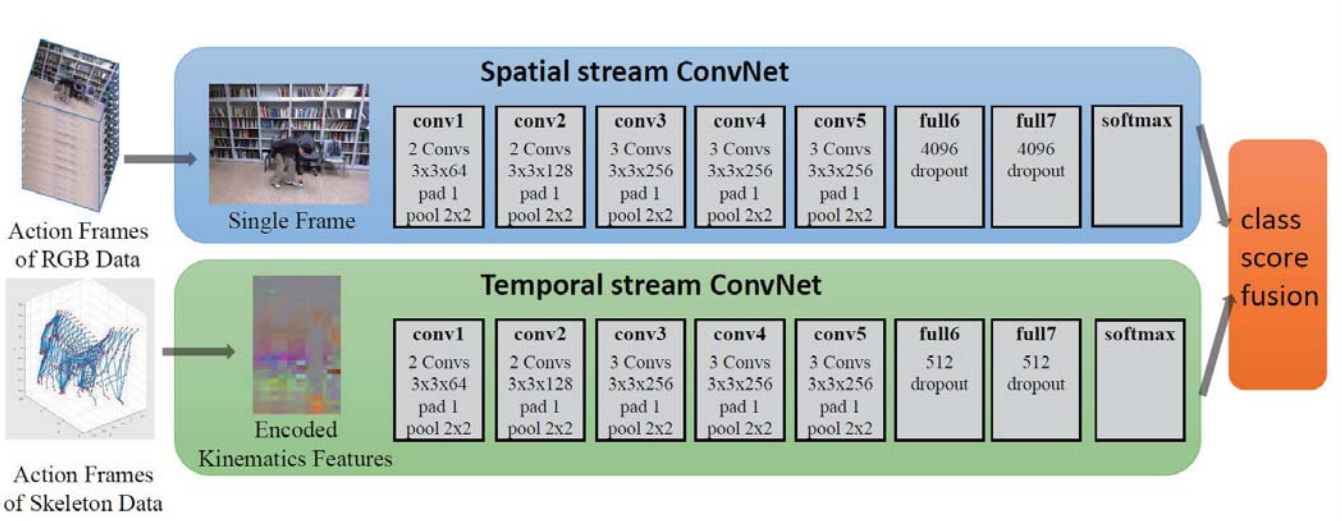

Fig. 4. Scheme of proposed two stream CNN based on RGB and skeleton data.

$(N-1)\times 20\times 3$ images is needed to achieve $32\times 32\times 3$ images.

The image transformation is processed in two steps and is operated on image rows $(N-1)$ and columns (20), respectively:

1. Key Frames Selection

Typically, an action consists of 40 60 frames, among which some of the frames are more informative. We try to select 32 key frames of each action, which not only well suits the expected input size, but also reduce the computational cost in training and testing. Given an action with frames N, linear velocity and orientation displacement can be computed into $(N-1)\times 20\times 3$ images. To select the key frames, we propose a energy function of row i described in the following:

$$\begin{aligned} E_i &= \sum_{j=1}^{20} \|v_i^j\|^2, i=[1;N-1], \quad or \\ E_i &= \sum_{j=1}^{20} \|\omega_i^j\|^2, i=[1;N-1], \end{aligned} \tag{7}$$

for linear velocity and orientation displacement, respectively.

Define the derivative of energy function with respect to frame i is $\delta E_i = E_{i+1} - E_i$, then frame i is more informative if the absolute value of δE_i is larger. Based on the absolute value of δE_i, we pick up the biggest 32 frames. One example is shown in Fig. 5.

2. Zero-padding

For images with 20 columns and less than 32 rows, we just pad the image left and right with 0 to achieve the expected input size.

The architecture of the VGG 16 and input size of spatial and temporal convnet in each layer is shown in the Table 1. By "/2", it means that after max-pooling layer, the input size is halved.

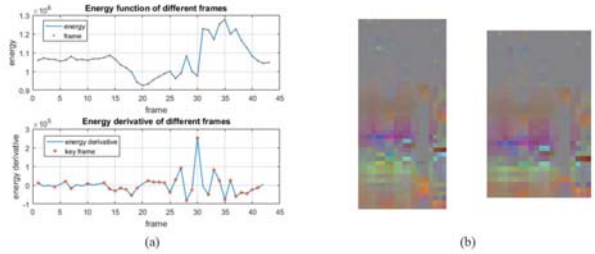

Fig. 5. Key frame selection of action *pick up* in the Northwestern-UCLA dataset.

IV. EXPERIMENT RESULTS

The proposed method is evaluated on North-western UCLA dataset [23]. This dataset contains RGB, depth and human skeleton data captured simultaneously by three kinect cameras. It includes 10 action categories and each action is performed by 10 actors, with around 150 samples per class. There are two types of protocols for evaluation of methods, cross-view and cross-subject. In cross-view protocol, view 1 and 2 are used for training an view 3 is for testing. In cross-subject protocol, subject 1 9 are used for training and subject 10 is for testing.

Pytorch was adopted as the CNN platform and two NVIDIA 1080Ti were used to run the experiment.

A. Performance of Encoded Kinematics Features

First, we measure the performance of our proposed encoded kinematics features, linear velocity and orientation displacement, individually. Both the cross-view and cross-subject protocols are considered. We also train on the spatial ConvNet on RGB frames, which will be used in the later fusion.

For the spatial ConvNet, we fine-tune the pretrained model VGG16, and subtract the mean image value of ImageNet dataset. For the temporal ConvNet, we train the VGG-16

TABLE I

INPUT SIZE OF SPATIAL AND TEMPORAL CONVNET OF EACH VGG16 LAYER

Model Architecture (VGG16)	Input Size (Spatial convNet)	Input Size (Temporal ConvNet)
3×3, 64 3×3, 64 2×2 max-pooling, /2	224×224×3	32×32×3
3×3, 128 3×3, 128 2×2 max-pooling, /2	112×112×64	16×16×64
3×3, 256 3×3, 256 3×3, 256 2×2 max-pooling, /2	56×56×128	8×8×128
3×3, 512 3×3, 512 3×3, 512 2×2 max-pooling, /2	28×28×256	4×4×256
3×3, 512 3×3, 512 3×3, 512 2×2 max-pooling, /2	14×14×512	2×2×512
full6, 128 full7, 128 softmax	7×7×512	1×1×512

like network from scratch, and subtract the mean image value of linear velocity data and orientation displacement data, respectively. In case of overfitting, we add two drop-out layer after each fully-connected layer with ratio 0.9. From the results, presented in Table 2, it is clear that our proposed encoded kinematics features, both linear velocity and orientation displacement, perform much better than the RGB frames. And as expected, the RGB images is more sensitive to view-change that in the cross-view protocol, its recognition accuracy is 3.2% less than the cross-subject protocol. But for our skeleton-based kinematics features, the accuracy of cross-view protocol even increased when compared with cross-subject one. And besides, the orientation displacement feature works better than the linear velocity, with 5% increase of the accuracy, although the later one has more physical meaning.

Note that the spatial ConvNet takes 8 hours to train with 2 1080ti GPU, while the temporal ConvNet only takes 20 minutes to train using the same GPUs.

B. Performance of Key Frame Selection

Next we evaluate our proposed key frame selection method. Two setting are considered: (1) use key frame selection method to select 32 more informative frames based on the derivative of the energy function, and (2) randomly select

TABLE II

DIFFERENT INPUT ACCURACY ON NORTHWESTERN-UCLA DATASET

Input configuration		Cross-subject	Cross-view
Encoded Kinematics Features	Linear Velocity	63.9%	64.4%
	Orientation Displacement	68.9%	69.5%
RGB frames		54.4%	51.2%

32 frames from the initial skeleton sequences. Performance of these two settings are presented in Table 3. We can tell from the results that the key frame selection method improves the performance of both cross-view and cross-subject protocol, but the improvement is not so remarkable when compared the improvement of temporal ConvNet to spatial ConvNet. This could because that normally, each action just contains 40 60 frames, so that randomly select 32 frames or delicately select will not make much difference.

TABLE III

PERFORMANCE OF ENCODED KINEMATICS FEATURES WITH AND WITHOUT KEY FRAME SELECTION

Input configuration	Key frame selection	
	off	on
Linear Velocity cross-subject	62.5%	63.9%
Linear Velocity cross-view	63.1%	64.4%
Orientation Displacement cross-subject	67.6%	68.9%
Orientation Displacement cross-view	68.4%	69.5%

C. Comparison with Handcrafted Features

Based on the results before, we finally fused the spatial ConvNet trained on RGB images and the temporal ConvNet trained on encoded orientation displacement features. We adopt the average fusion method described in two stream CNN [2]. And the comparison between our method and other popular handcrafted features is shown in the following table. Our proposed method beats most of the handcrafted features, such as CVP, HON4D and SNV.

TABLE IV

COMPARISON OF DIFFERENT HANDCRAFTED FEATURES WITH OUR METHOD

Method	Recognition accuracy
CCD [24]	34.4%
DVV [25]	52.1%
CVP [26]	53.5%
HON4D [27]	39.9%
SNV [28]	42.9%
Our method	**75.6%**

V. CONCLUSIONS

We have presented a novel framework to combine the traditional handcrafted feature method with deep learning method. Kinematics features, including linear velocity and orientation displacement, are proposed to capture the temporal information of human action. We have introduced a

novel image encoding method to allow CNN training on the proposed kinematics features. Key frame selection is proposed to guarantee the same input size of CNN. We evaluate our method on a popular and challenging multi-view data and the experiment results show that our proposed method is fast to train and outperforms many handcrafted features.

REFERENCES

[1] I. Laptev, On space-time interest points, International journal of computer vision, 2005, 64(2-3), pp. 107-123.

[2] K. Simonyan and A. Zisserman, Two-stream convolutional networks for action recognition in videos, Advances in neural information processing systems, 2014, pp. 568-576.

[3] C. Wang, Y. Wang, and A. L. Yuille, Mining 3D key-pose-motifs for action recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2639-2647.

[4] A. Gilbert, J. Illingworth, and R. Bowden, Scale invariant action recognition using compound features mined from dense spatio-temporal corners, European Conference on Computer Vision, Springer, Berlin, Heidelberg, 2008, pp. 222-233.

[5] P. Scovanner, S. Ali, and M. Shah, A 3-dimensional sift descriptor and its application to action recognition, Proceedings of the 15th ACM international conference on Multimedia, ACM, 2007, pp. 357-360.

[6] G. Willems, T. Tuytelaars, and L. Van Gool, An efficient dense and scale-invariant spatio-temporal interest point detector, European conference on computer vision, Springer, Berlin, Heidelberg, 2008, pp. 650-663.

[7] H. Wang, A. Klser, Schmid C, and C. L. Liu, Action recognition by dense trajectories, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 3169-3176.

[8] H. Wang and C. Schmid, Action recognition with improved trajectories, Proceedings of the IEEE Conference on Computer Vision, 2013, pp. 3551-3558.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems, 2012, pp. 1097-1105.

[10] R. Girshick, J. Donahue, T. Darrell and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580-587.

[11] S.Ji, W. Xu, M. Yang, and K. Yu, 3D convolutional neural networks for human action recognition, IEEE transactions on pattern analysis and machine intelligence, 2013, 35(1), pp. 221-231.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, Large-scale video classification with convolutional neural networks, Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2014, pp. 1725-1732.

[13] D. Tran, L. D. Bourdev, R. Fergus,L. Torresani, and M. Paluri, C3D: generic features for video analysis. CoRR, abs/1412.0767, 2(7).

[14] C. Feichtenhofer, A. Pinz, and A. Zisserman, Convolutional Two-Stream Network Fusion for Video Action Recognition, Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 1933-1941.

[15] G. Gkioxari and J. Malik, Finding action tubes, Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 759-768.

[16] J. Shotton, A. Fitzgibbon, and M. Cook, Real-time human pose recognition in parts from single depth images, Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2011, pp. 1297-1304.

[17] j. Liu, A. Shahroudy, D. Xu, A. K. Chichung, and G. Wang, Skeleton-based action recognition using spatio-temporal lstm network with trust gates, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[18] V. Veeriah, N. Zhuang, and G. J. Qi, Differential recurrent neural networks for action recognition, Proceedings of the IEEE Conference on Computer Vision, 2015, pp. 4041-4049.

[19] W. Zhu, C. Lan, and J. Xing, Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks,AAAI, 2016, 2: 8.

[20] N. Srivastava, G. Hinton, and A. Krizhevsky, Dropout: A simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research, 2014, 15(1), pp. 1929-1958.

[21] A. Krizhevsky and G. Hinto, Learning multiple layers of features from tiny images, 2009.

[22] https://github.com/chengyangfu/pytorch-vgg-cifar10

[23] J. Wang, X. Nie, and Y. Xia Y, Cross-view action modeling, learning and recognition[C], Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2649-2656.

[24] Z. Cheng, L. Qin, and Y. Ye, Human daily action analysis with multi-view and color-depth data, European Conference on Computer Vision, Springer, Berlin, Heidelberg, 2012, pp. 52-61.

[25] H. Rahmani, A. Mahmood and D. Huynh, Action classification with locality-constrained linear coding, Proceedings of the IEEE conference on pattern recognition, 2014, pp. 3511-3516.

[26] Z. Zhang, C. Wang, and B. Xiao, Cross-view action recognition via a continuous virtual path, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2690-2697.

[27] O. Oreifej and Z. Liu, Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences, Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 716-723.

[28] X. Yang and Y. L. Tian, Super normal vector for activity recognition using depth sequences, Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 804-811.