3D Human Skeleton Data Compression for Action Recognition

Sheng Li, Tingting Jiang*, Yonghong Tian, Tiejun Huang
NELVT, Department of Computer Science, Peking University, China

***Abstract*—Skeleton-based action recognition continues to open up new application scenarios with the popularity of acquisition devices. This also leads to a rapid increase in the amount of human skeleton data. Currently, there is no skeleton data compression algorithm for the task of action recognition. In order to solve this problem, we propose the first skeleton data compression algorithm, which can compress the skeleton data stream to a small bandwidth while keeping the accuracy of action recognition as high as possible. The proposed compression algorithm is called Motion-based Joints Selection (MJS). It performs compression based on the amount of movement of different joints. In addition, we also explored the combination of MJS and existing lossless compression methods, and found the most suitable one. In the end, we verify that our compression method MJS can achieve promising results on the large dataset NTU-RGB+D.**

***Index Terms*—Human Skeleton Data, Data Compression, Action Recognition**

I. Introduction

Human pose, also known as skeleton, can be used as a kind of data modality for action recognition. Skeleton-based action analysis is more efficient and less noisy than video-based action analysis. In general, the 'skeleton data' mentioned is a skeleton sequence. As shown in Figure 1 (a), skeleton data is essentially a matrix with size of $Time \times Joint$.

Advances in acquisition technology have greatly widened the use of 3D skeleton-based action recognition. Such as, sports guidance, autopilot, mobile robots, HCI (Human-Computer Interaction) and surveillance. With the popularity of depth cameras, the acquisition of human skeleton data is becoming more and more convenient. Sensors such as Microsoft Kinect [1] and Intel RealSense [2] have entered the daily lives of many people. And some long distance depth cameras are being installed in smart cars and security systems. This has led to a very rapid increase in the amount of human skeleton data. On the other hand, with the popularity of network infrastructure and the development of cloud computing. The demand for skeleton data transmission is also increasing. Especially in the Internet of Things era, sensors are only responsible for data collection and transmission, and the analysis is done by the cloud. For example, in a crowded large event with 10,000 people, 5 cameras may produce 50,000 samples of skeleton sequences. If each sample has 25 joint points of 25 frames per second, the amount of data generated is up to 3000 Mbit/sec. Through this example, it can be seen that in a crowded scene, skeleton data may put great pressure on network transmission. Therefore, we have to consider the compression problem of the skeleton data. This is what we are trying to solve.

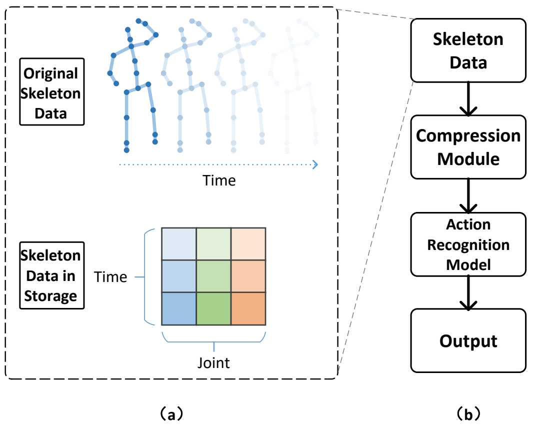

Fig. 1. (a) shows the form of skeleton data in the real world and in computer memory. In (b), the compression module is our proposed compression algorithm. It shows the stage where the compression algorithm (compression module) works.

Skeleton data has certain similarities with point cloud data, both of which are floating point numbers and exist in three-dimensional space. It must be said that compared to the skeleton data, the amount of data in the point cloud is much more. So the compression requirements of point cloud data have appeared since a long time ago. Unfortunately, although the point cloud compression algorithm has been greatly developed, it is not suitable for skeleton data compression. Because the point cloud compression algorithm is designed for dense points. For example, the B-spline [3] method can represent a type of methods, which keep more points in the location with greater curvature to preserve detail. For human skeleton data with sparse joint points, this method is not applicable. On the other hand, the purpose of our compression is not to reconstruct data, but to satisfy action recognition.

To the best of our knowledge, there is currently no skeleton data compression algorithm dedicated for action recognition. There is not even a general compression algorithm designed for human skeleton data. For skeleton action recognition, not every joint point is critical to the discrimination. This

* Corresponding author. This work was partially supported by National Basic Research Program of China (973 Program) under contract 2015CB351803 and the Natural Science Foundation of China under contracts 61572042, 61527804. We acknowledge the high-performance computing platform of Peking University for providing computational resources.

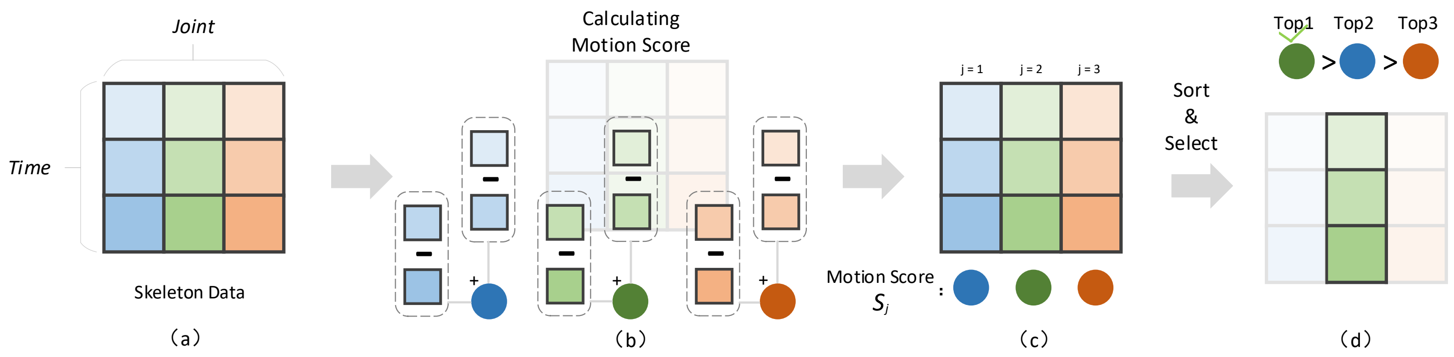

Fig. 2. The key steps of the MJS algorithm. (a) is the original skeleton data. In the step (b), the motion flow of the skeleton data is calculated, and the motion intensity of each joint is calculated, that is, Motion Score. (c) is the Motion Score for each joint point. In the step (d), the top K joints are selected according to the score, and their joint ID numbers are recorded.

is similar to human perception of action. When we judge a person's action, it is not necessary to see all the joints of this person. For example, for the action of clapping, as long as we see the trajectory of the joints (hands), we can make a judgment. At this point we don't care where the joint of the head is. Psychological experiments [4] have also confirmed this. Many methods of skeleton-based action recognition take advantage of this and introduce many techniques such as attention mechanism [5, 6] and part-aware mechanism [7] to exploit them. We are applying such a 'selection mechanism' to the compression of skeleton data in action recognition. We propose a novel and efficient skeleton data compression algorithm for action recognition which is called Motion-based Joints Selection (MJS). The method is based on a selection strategy. In addition, we have designed a dedicated serialization method to ensure that the processed data can be well combined with the lossless compression algorithms. The stage of the compression process in the skeleton action recognition is shown in Figure 1 (b). By the way, our approach is also compatible with existing skeleton action recognition models. We evaluated our approach with multiple metrics on the large dataset NTU-RGB+D [7] and it performs very well.

The rest of the paper is organized as follows. In Section II, we introduce the proposed method. In Section III, experimental results of our method is described in detail. In Section IV, we draw a conclusion and clarify the future work.

II. Method

This section contains two parts. The first part is the method we propose, and the second part is the combination of our method and lossless compression algorithms.

A. Motion-based Joints Selection

The skeleton data of a person at time step t can be denoted as $X_t = \{X_t^1, X_t^2, ..., X_t^j, ..., X_t^J\}$ where X_t^j refers to the coordinate value of the joint j at time t and J is the number of joints. For a complete skeleton sequence, it can be formulated as $X = \{X_1, X_2, ..., X_t, ..., X_T\} \in \mathbb{R}^{T \times C \times J}$, where T is the number of frames in the sequence and C is the dimensionality of the coordinate space. For 3D skeleton data, $C = 3$.

Figure 2 (a) is the skeleton data after omitting the channel dimension. In general, joints with a large amplitude of motion have a great impact on the discrimination of action. Our compression algorithm takes advantage of this. This is Motion-based Joints Selection (MJS). The first step is the calculation of the motion of each joint. The skeleton motion is defined as the temporal difference of each joint between two consecutive frames as M_t in Eq. (1).

$$\begin{aligned} M_t &= X_{t+1} - X_t \\ &= \{X_{t+1}^1 - X_t^1, X_{t+1}^2 - X_t^2, ..., X_{t+1}^J - X_t^J\} \end{aligned} \tag{1}$$

The motion of a skeleton data sequence can be written as M, and M is $\{M_1, M_2, ..., M_{T-1}\}$. Obviously, $M \in \mathbb{R}^{(T-1) \times C \times J}$. Next, as shown in Figure 2 (b), we need to calculate a motion score for each joint. For the score of joint j we can record it as S_j. Its calculation is as in Eq. (2).

$$S_j = \sum_{c=1}^{C} \sum_{t=1}^{T-1} |M_{t,c,j}| \tag{2}$$

S_j represents the intensity of the motion amplitude of the joint j. Finally, as shown in Figure 2 (d), the S_j of all joint points are sorted, and the top K joints with the largest S_j value are selected. These are the joints we want to keep. By adjusting the value of K, we can easily control the compression ratio. It should be noted that during the transmission, we need to transmit the ID numbers corresponding to the K joints which are used to distinguish different joints. This is crucial for the 'recovery' after the transmission.

After transmitting the compressed data, it cannot be directly sent into the model but requires simple 'recovery'. That is, the K joint points that have been transmitted need to be placed at the original position according to their ID numbers, and the remaining $(J - K)$ joint points are directly filled with 0. At this point, the skeleton data with a lot of 0 can be directly sent to the action recognition model to complete the analysis. It should be noted that when training the model, it is necessary to use the data compressed by the MJS algorithm. This will ensure that the algorithm works best.

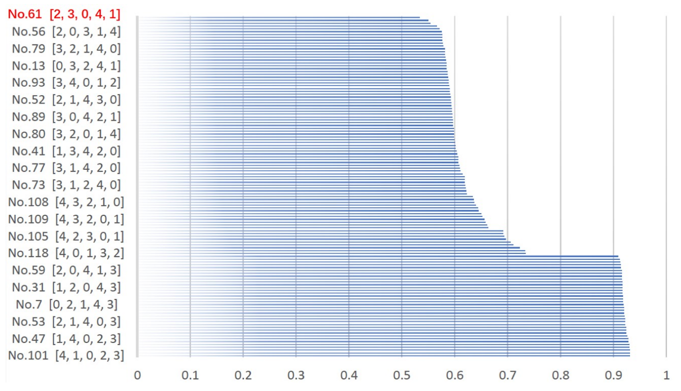

Fig. 3. Compression ratio in different serialization modes. The horizontal axis represents the compression ratio, and the vertical axis represents the different dimensional order, where 0, 1, 2, 3, and 4 represent the dimensions of N, C, T, J, and P, respectively.

B. *Combination with lossless compression algorithms*

After the skeleton data was compressesed by MJS algorithm, we can continue to use the general lossless compression algorithm to further improve the compression ratio. For explanation, we extend the definition of X in the previous section. The deep learning method has already dominated the skeleton-based action recognition. There are numerous methods based on RNN, CNN, and GNN in recent years. No matter which deep learning model is adopted , when the model is used to infer the action class of their samples, the data stream can be regarded as a 5-dimensional tensor expanded from 3-dimensional X just mentioned in Section II-A. Here we can record it as $X' \in \mathbb{R}^{N \times T \times C \times J \times P}$. For the extra two dimensions, N is the size of the batch size and P is the number of people included in the sample.

Lossless compression algorithms have been developed over forty years. DEFLATE (RFC1951) [8] has become one of the most important algorithms and is widely used in applications such as PNG, ZIP, GZIP, etc. Here, we combine MJS with the DEFLATE algorithm. Serialization is the process of converting a data object into a stream that can be transferred or stored. Before the data is compressed by the DEFLATE algorithm, it must be serialized. For a tensor form of data, by adjusting the order of the dimensions, there are a variety of serialization methods, such as X' here. However, due to the limited size of the sliding window in the DEFLATE algorithm, even if the same raw data is used, there may be a significant difference in compression effect due to the use of different serialization methods. For the 5-dimensional X, even if serialized by sequential scanning, there are 5!=120 ways. Through testing on the NTU-RGB+D dataset, we found the best among the 120 ways. That is to serialize according to the dimensional order of $T-J-N-P-C$. The results of all combinations are shown in Figure 3. It can be seen that serialization according to different dimensional order, the final compression ratio of the DEFLATE algorithm is significantly different. The compression ratio spans from 64.5% to 93.2%. The serialization methods which rank the T dimension at the first one have a very good compression ratio, which is also very reasonable. Because if you put two adjacent frames together, for example $X'_{t,j,n,p,c}$ and$X'_{t+1,j,n,p,c}$, the probability that the two values are exactly the same will be much higher. This is easier to compress by the DEFLATE algorithm.

III. Experiment and Analysis

In this section, we will introduce the experiment, including the two main parts of the experimental settings and experimental results analysis.

A. *Experimental Settings*

To evaluate the effectiveness of the proposed method, we performed experiments on the skeleton-based action recognition dataset. Because our compression algorithm is used for action recognition, the change of final accuracy rate is a measure of the pros and cons of the algorithm, while the general compression algorithm needs to pay attention to the error rate after data recovery. The NTU-RGB+D [7] is currently the most widely used skeleton-based action recognition dataset. We choose cross-view (CV) protocol as the final evaluation protocol here. Under this protocol, the training data comes from cameras at view 2 and 3, while the data from cameras with view 1 is used for testing.

For the convenience of description, A_1 and A_2 are introduced here to indicate the action recognition accuracy and the accuracy after MJS compression (the DEFLATE algorithm is lossless and does not affect the accuracy). D_1, D_2, and D_3 represent the amount of original data, the amount of data after lossy compression (methods such as MJS), and the amount of data after further lossless compression algorithm (DEFLATE). $\alpha = A_2/A_1$ indicates the retention of the accuracy after compression. $\beta_1 = D_2/D_1, \beta_2 = D_3/D_1$ respectively represent the ratio of the current data size to the original data size.

HCN [9] model is considered as the baseline for skeleton-based action recognition which is a state-of-the-art CNN-based model. It makes full use of CNN's global modeling ability in the channel dimension, and strengthens the model's global modeling ability for joint points by replacing the joint dimension and the channel dimension. Although the model is not open source, we have reimplemented it. And under the CV protocol, the accuracy of the HCN model reached 91.1% which is A_1 defined above.

B. *Experiment analysis*

One advantage of the MJS method is that the compression ratio is very easy to control. Figure 4 shows the results of our experiments in a comprehensive way. 100% represents the original amount of data, where the yellow and blue rectangles represent the $1-\beta_1$ and $\beta_1-\beta_2$, respectively. The green rectangle is the amount of data remained in the end that is β_2. For that dark blue curve, the vertical axis refers to the value of α. The horizontal axis refers to the number of joint points retained in the MJS algorithm. This clearly shows the effect of the MJS and DEFLATE algorithms on the compression ratio. It can be found that when 11 joints are retained, α is 91.2% ,

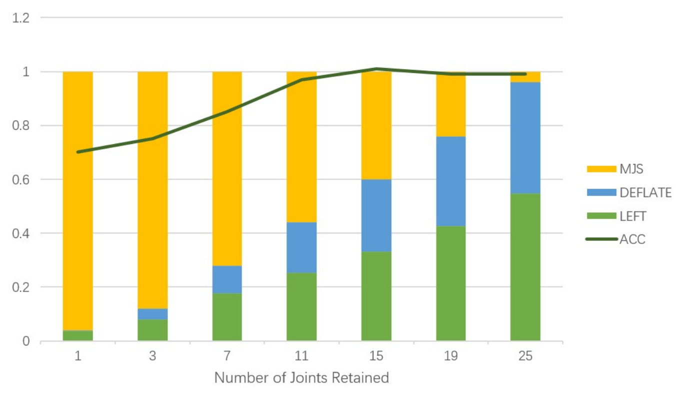

Fig. 4. The relationship between the number of joints retained, the compression ratio, and the recognition accuracy. The yellow part is the amount of data that is reduced by the MJS algorithm $(1-\beta_1)$. The blue part is the amount of data that is reduced by the DEFLATE algorithm $(\beta_1-\beta_2)$. The green part is left after two compressions(β_2). The dark blue curve indicates the value of α. The smaller the compression amplitude, the higher α.

but at this time the amount of data is reduced by 73.5%. It is worth noting that when using the MJS algorithm to preserve 15 joint points, the accuracy of the model even exceeds the state when it is not compressed. We think that this is mainly because the compression process of the MJS algorithm can reduce the noise of the data to a certain extent, which leads to the performance improvement.

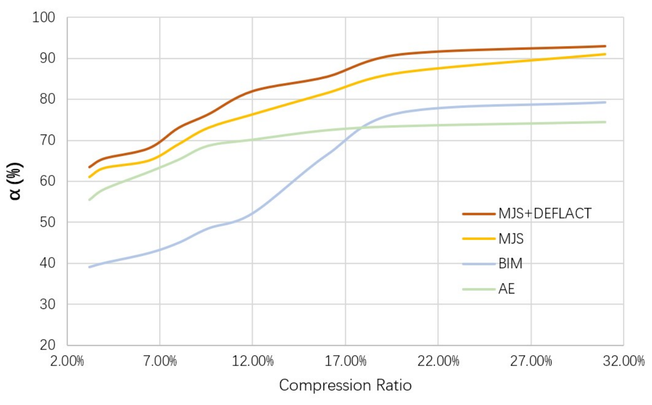

Fig. 5. The curves drawn by the values of α are different under several methods. In the case of the same compression ratio, the red one MJS+DEFLATE has the highest α. For each method, as the compression ratio increases, the value of α increases.

We also did a comparative experiment with other methods. We have chosen the two basic methods for compression, which are Autoencoder (AE) [10] and bidirectional linear interpolation method (BIM) [11]. It can be seen from Figure 5 that the MJS+DEFLATE method has better final accuracy (α) than these two at the same compression ratio. Even if the MJS algorithm is used alone, it still performs very well.

At the same time, we also compared the gain of DEFLATE based on different methods. The comparison results are shown in Figure 6. The results show that the MJS method has the best compatibility with the DEFLATE algorithm. The compression is most effective when the two are combined.

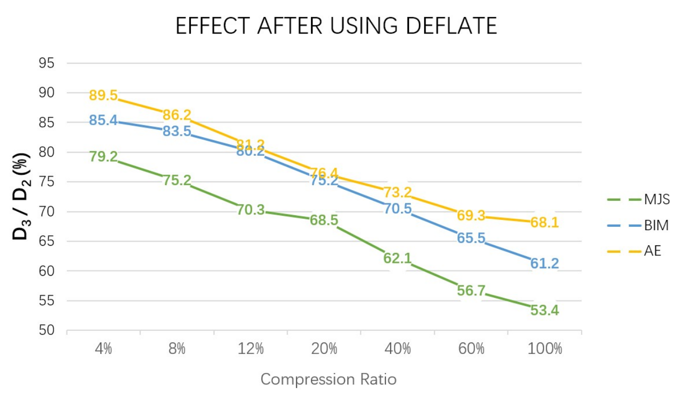

Fig. 6. Gain comparison after adding the DEFLATE algorithm. The horizontal axis represents different compression ratios, and The vertical axis indicates the D_3/D_2 (less is better). The DEFLATE algorithm brings the most notable gain to the MJS method. That is the combination of the two is more effective than others.

Conclusion

In this paper, we proposed the method called MJS which is the first compression algorithm for skeleton data in action recognition. We also verified the effectiveness of the proposed method on the NTU-RGB+D dataset. Under the same compression ratio, The MJS algorithm has the excellent performance compared to other methods. At the same time, we also explored the combination of the MJS method and the traditional lossless compression algorithm, and found an effective way. Even so, we still have some work to do in the future. For example, one of them is that in a multi-person scenario, how to use the redundant information of different people to further improve the compression ratio.

References

[1] Zhengyou Zhang. Microsoft Kinect sensor and its effect. *IEEE Multimedia*, 19(2):4–10, 2012.

[2] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel RealSense stereoscopic depth cameras. *arXiv preprint arXiv:1705.05548*, 2017.

[3] Eric Saux and Marc Daniel. Data reduction of polygonal curves using B-splines. *Computer-aided design*, 31(8):507–515, 1999.

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

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[6] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[7] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. NTU RGB+ D: A large scale dataset for 3D human activity analysis. In *CVPR*, pages 1010–1019, 2016.

[8] Peter Deutsch. Deflate compressed data format specification version 1.3. Technical report, 1996.

[9] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation. *arXiv preprint arXiv:1804.06055*, 2018.

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Autoencoder *Deep Learning*, volume 14, pages 499–504 ,2016

[11] Steven Kay. Some results in linear interpolation theory *IEEE Transactions on Acoustics, Speech, and Signal Processing*, volume 31, pages 746–749 ,1983