4D Hand Gesture Recognition using Point Set

Sora Ryu
sryu@umass.edu

Xiangdong Xie
xxie@umass.edu

Abstract

In this paper, we propose our own 4D Hand Gesture Recognition approach based on PointNet and LSTM. The input of the method is the depth frames from the short RGB-D video of gesture, we then generate point cloud for each frame, use PointNet to extract global feature of sequence, and then use LSTM to grasp temporal relationship of data and classify the gesture. Our model uses only depth images, no skeletal data or joints.

1. Introduction

Gesture is one of our effective and intuitive body languages where we can successfully communicate and convey the semantic meanings. In that sense, hand gesture recognition task has grown its importance in the computer vision domain. Gesture recognition has been actively researched in the application of Human-Computer Interaction, where intuitive and easy-to-use interfaces are needed.

Thanks to the rising performance of sensor depth camera, including Microsoft Kinect, Leap Motion and Intel RealSense, we are able to easily obtain the high quality of depth and skeletal data. Many recent studies leverage those data for the tasks including gesture classification, hand tracking, pose estimation and so on.

Recent skeleton based gesture classification approaches [2, 9, 1] basically rely on the hand joint information returned by the sensor depth camera to extract powerful and effective hand descriptor. Devineau et al [3] proposes the approach of classifying hand gesture from the 3D skeletal data using CNNs. This approach extracts relevant features from the given 3D skeletal data sequence by using 1D CNN for each channel(x,y,z axis), and then take the merged features as an input for dense neural network to classify its gesture. Skeleton-based methods are robust to varying lighting conditions, but the performance highly depend on the quality of estimated results.

Lai et al [6] leverages both depth and skeleton data for hand gesture recognition, where the method is consisted of a depth-based CNN+RNN and a skeleton-based RNN. Mahmud et al [8] proposes Fusion Model architecture which concatenates 2D skeletal joints and depth sequence for gesture prediction. However, those approaches have some limitations considering that it is expensive to use both data.

Considering the above issues, we propose our method which is much simpler and less expensive in that it only utilizes depth frames but preserves the high performance. To be specific, we convert each frame into 3D point clouds, and develop a model which classify the gesture of sequence based on 4D point cloud data.

2. Related Works

PointNet[10] and PointNet++[11] are the benchmarks for using point clouds for various types of tasks, including classification and segmentation. PointNet directly consumes raw point cloud as an input, and outputs either class labels for entire point set or part labels for each point. PointNet++ does hierarchical feature learning via repeating sampling and PointNet and takes account into the normals of the point cloud, resulting in a better performance than PointNet.

Salami et al [12] proposed the PointGest which directly uses 4D point clouds to recognize the motion pattern. The model is consisted of PointNet++ and LSTM where each processes 3D point cloud and extractes temporal features of a 4D point cloud.

PointLSTM [9] proposed by Min et al, is the model which can directly capture temporal relationships from dynamic point cloud sequences via LSTM layer. Inspired by this paper, we got an idea of using LSTM for processing 4D point cloud sequence.

Liang et al [7] proposed a multi-view method that recognizes the hand gestures using point cloud. The projection of point cloud into multi-view images and extracting and fusing the features from those images were the main idea of the paper.

3. Method

An overview of our method is shown in Figure 1. We generate 3D point clouds for each frame, and do the data preprocessing including point cloud sampling, frame sampling and statistical outlier removal to create a sequence of point clouds. After we build our dataset, we feed the point cloud data to a model which consists of PointNet and LSTM and classify its gesture label.

3.1. Dataset

We used Dynamic Hand Gesture (DHG) dataset [13] which consists of sequences for 14 different hand gestures, including grab, tap, expand, pinch, swipe and so on. The gestures are performed in two ways, using one finger or the whole hand, but we only used the latter one, and thus 1400 sequences are used. Each sequence contains the depth frame at each time step as well as the 2D and 3D skeletal data where the coordinates of 22 hand joints are described. However, we didn't use any skeletal information here and solely used depth frames. The list of gestures in the DHG dataset is shown in Table 1

Gesture	Number of sequences
Grab	100
Tap	100
Expand	100
Pinch	100
Rotation Clockwise	100
Rotation Counter Clockwise	100
Swipe Right	100
Swipe Left	100
Swipe Up	100
Swipe Down	100
Swipe X	100
Swipe V	100
Swipe +	100
Shake	100

Table 1. The list of the 14 gestures in the Dynamic Hand Gesture (DHG) dataset.

3.2. Data Preprocessing

In this section, we introduce the framework of how we processed DHG dataset [13] into the dataset for our model where point clouds are needed.

3.2.1 Point Cloud Generation

First, we generate 3D point cloud for each frame. The DHG dataset depth images include not only hand but also face and torso. As our region of interest (ROI) is hand, we need to segment hand from depth images based on the depth value threshold. We have empirically found that the 680 would be the best threshold value for the segmentation. Then, we create 3D point cloud from each segmented hand. The example result of point cloud generation is shown in Figure 2

3.2.2 Point Cloud Sampling

Each frame has 15000 points on average, which results in tremendous amount of points for sequence in total. Therefore, we considered two different point cloud sampling methods: Farthest Point Strategy (FPS) [4] and Random Point Sampling. FPS samples the points via adding the farthest from the current set of sample points at each stage. The algorithm calculates the minimum distances between remaining points and sampled points, and sample the point which has the maximum distance. So, we can sample points which is the farthest point from all sampled points, resulting in uniform sampling effect. However, FPS needs a lot of computation for computing every distance between remaining points and sampled points. As each frame generates over 10000 points and each sequence contains 80 frames on average, it took so long time for sampling via Farthest Point Strategy for all 1400 sequences. Therefore, we end up choosing random sampling which enables faster processing.

For random sampling method, we uniformly downsample point cloud by collecting every 20th points and remove outliers which are far away from their 20 neighbors compared to the average for the point cloud with standard deviation ratio of 0.3. Then, we randomly pick 600 points, and hence, 600 3D point clouds are the representative of a single depth frame. The comparison between two sampling methods is shown in Figure 3. We can see that the FPS samples the points more stably and uniformly, but unfortunately we were not able to use the algorithm because of the limited computing resource.

3.2.3 Frame Sampling

Even though we decreased the number of points, we are still having a lot of points considering in a sequence view. Also, the points of consecutive frames do not differ a lot, and we needed to fix the size of frames as the sequence lengths varied. Considering the above issue, we also tried frame sampling, where we randomly sample the frames into 20 frames, resulting in a fixed length of sequence as 20.

3.3. Build Dataset for Our Model

After we complete data preprocessing, we are able to get point clouds with the shape of (20, 600, 3). The valid sequences were 1398, and thus, the dataset has the shape of (1398, 20, 600, 3), where 1398 refers to the number of sequences, 20 refers to the number of frames, 600 refers to the number of points, and 3 refers to the point cloud dimension. We split the dataset into training set and test set with 70:30 ratio, resulting in (978, 20, 600, 3) and (420, 20, 600, 3) tensors respectively.

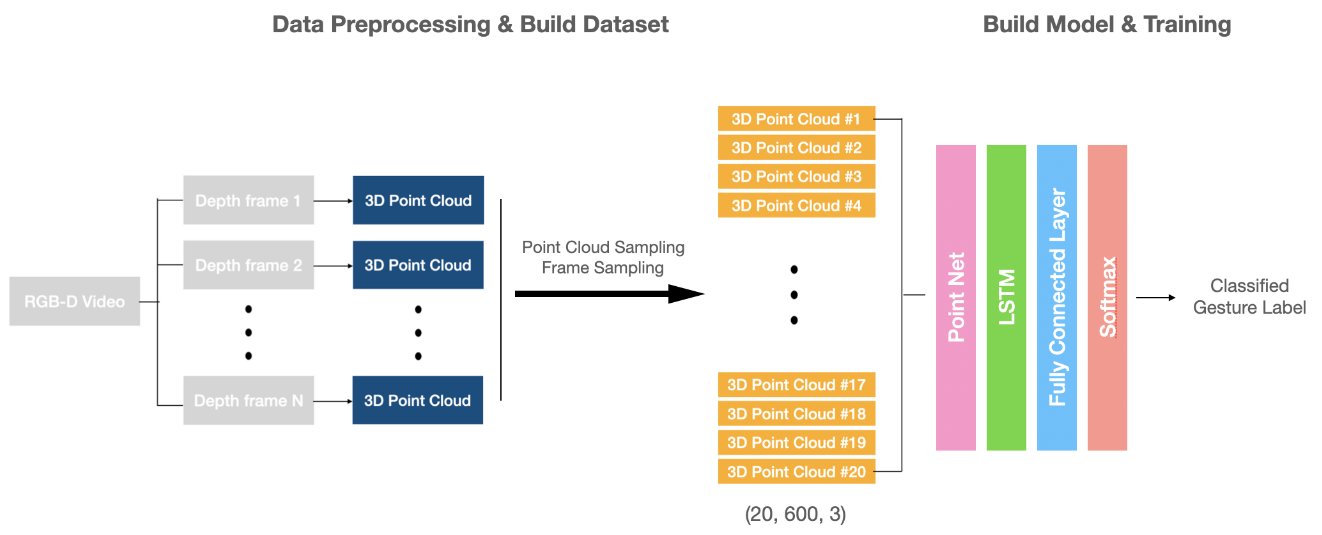

Figure 1. Overview of our approach. We do not use any skeletal data. We have fixed the number of point clouds for each frame and the number of frames into 600 and 20 respectively. Hence, the data for each sequence can be represented into point clouds with the shape of (20, 600, 3).

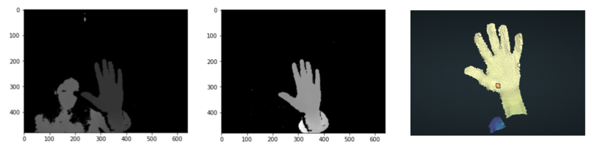

Figure 2. Point cloud generation. The image are original depth image, hand segmentation and point cloud generation, respectively.

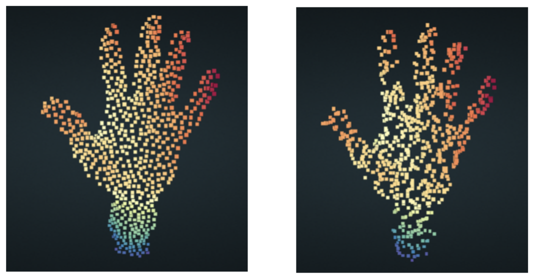

Figure 3. The comparison between two sampling methods. Left image shows the FPS algorithm and right image shows the random sampling method. Both are sampled with 600 points.

3.4. Model Architecture

We introduce a PointNet [10] and Long short-term memory (LSTM) based model which are able to classify the gesture of its input sequence. The idea is to extract the spatial features out of each frame of the motion using the same model with the same parameters, and try to find the temporal pattern of the gesture to get a probability distribution of the final classification. The whole model architecture is depicted in Figure 5. The model originally uses simple RNN, with some experiments and the inspiration from PointGest in [12], we changed to LSTM as the short term memory of RNN may lead to a severe under-learning of the Point Net before the RNN.

The Point Net part of our model is described in Figure 4. For each frame, a vector of global features of the input point cloud is extracted by the PointNet module. Multiple Point Nets with shared weights are used to to generate a sequence of global features. Then, those features are fed into the LSTM with hidden state of size H. At the end there are two fully connected layers which have dropout layer in between, and the model outputs the raw scores over 14 classes. Finally we use the cross entropy loss as our criterion and back-propagate the model with it.

3.5. Training

The default hyper-parameter values we use are: learning rate= $1e^{-2}$; weight decay= $1e^{-5}$; global feature number= 64; hidden layer number in LSTM= 2; After setting these hyper-parameters, we train the model using Adam algorithms in 100 epochs with a batch size of 32.
The initial training failed, as the model never learns, even after 1000 epochs, changing optimizing algorithms and weight decay does not help either. Our guess was that we

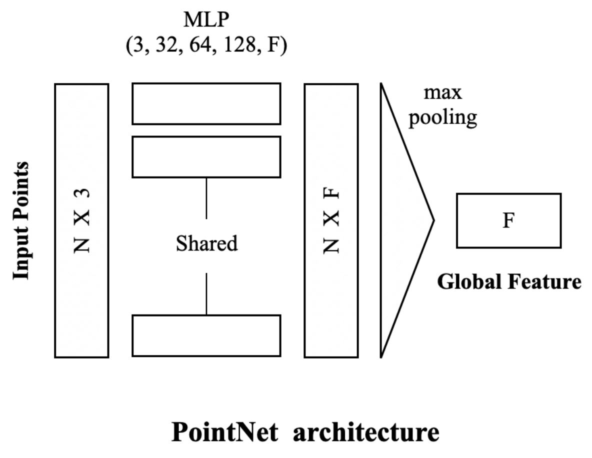

Figure 4. The architecture of PointNet. The model takes as input a set of points (Nx3) represented by 3D coordinates, and outputs a global feature as a descriptor of feature representation for the input points. N is the number of points, and F is the number of output features.

lack a supervised feature extraction learning phase, and we initially use RNN instead of LSTM, so the PointNet part of the model will never learn, therefore the RNN never gets a meaningful/structured input feature tensor that represents the raw gesture data well enough. We then changed RNN to LSTM, and it failed again. After some experiments, we decided to, though with no mathematical correctness (actually a totally different function mathematically), skip the LSTM part altogether and train the model with only the spatial feature extraction part along with the fully connected layer at the end of it. And this worked, as the model was then able to learn something, pushing its training accuracy and validation accuracy to around 80% and 65%, respectively. Finally we add LSTM back in and the model learns better, but had over-fitting. In the experiment we are exploring the best hyper-parameter values and settings for the model. Additionally, we changed the optimizer to use SGD with nesterov momentum because it has better generalization performance.

4. Experiment

After realizing that the model must start learning without LSTM, we design a 2 part training: first train the model without LSTM for 100 epochs and then train it with LSTM for another 100 epochs. We tried some combinations of different hyper-parameter values, the results are shown in the graphs 6:

First, we found that the weight decay should be smaller than somewhere around $= 1e^{-3}$, otherwise the model just stops learning. Also the more layers we have in the Point Net and more features we are trying to extract, within a reasonable range (512 is enough), the better the model generalizes, but just by around 2% and is not significant. The number of hidden layers in LSTM (we tested 1, 2 and 3) does not seem to matter whatsoever. Adding dropout layers between the linear layers in the MLP of the Point Net significantly slows down the computation, with insignificant increase in performance (around 1%)

5. Conclusion

Due to the lack of joint locations, which are ground-truth descriptors of some of the points in the point cloud, our model cannot learn the key features of a hand under supervision. This is very unfortunate, but we somewhat get around this problem by letting the feature extraction layers, the Point Net module, learn some of the features by concatenating it directly with the FC layer following our LSTM module and output classification scores. Last we put LSTM back in the model to better extract the temporal features of the motion.
With some experiments, we discovered that the weight decay should stay around 0.001 to make the model generalize best while still being able to learn. Layer numbers and feature size do not seem to affect the performance of our model significantly, which means that our model can be really simple. In fact, our model only has ninety thousand parameters if we choose to implement 2 hidden layers and extract 32 global features.

6. Discussion

Our initial approach was to implement a 2-part model for the task which is divided into joint location estimation and gesture classification sub tasks. We tried to first extract a point cloud from every single 2D depth frame from the RGB-D video[5], then train the model to locate the relevant hand joints[3] in the given point cloud, and finally train the CNN model to generate a classification of the gesture using the previously trained joint location estimator. However, the problem was that we couldn't find the estimated joint location from the generated point cloud, as the coordinates of generated point cloud and ground truth skeletal joints were different. So, we were not allowed to directly use the joint coordinates into the point cloud, and point cloud alignment between two point clouds were needed. However, as the number of points are different, where skeletal joint data has 22 points and generated point cloud has about 20000 points, we couldn't do the point cloud alignment to match the coordinates. Hence, we changed our approach to directly classify gesture without the joint estimation.

As for the possible improvement of the model, we believe that it is the Point Net that cannot extract the features well enough, since RNN in general will make modules be-

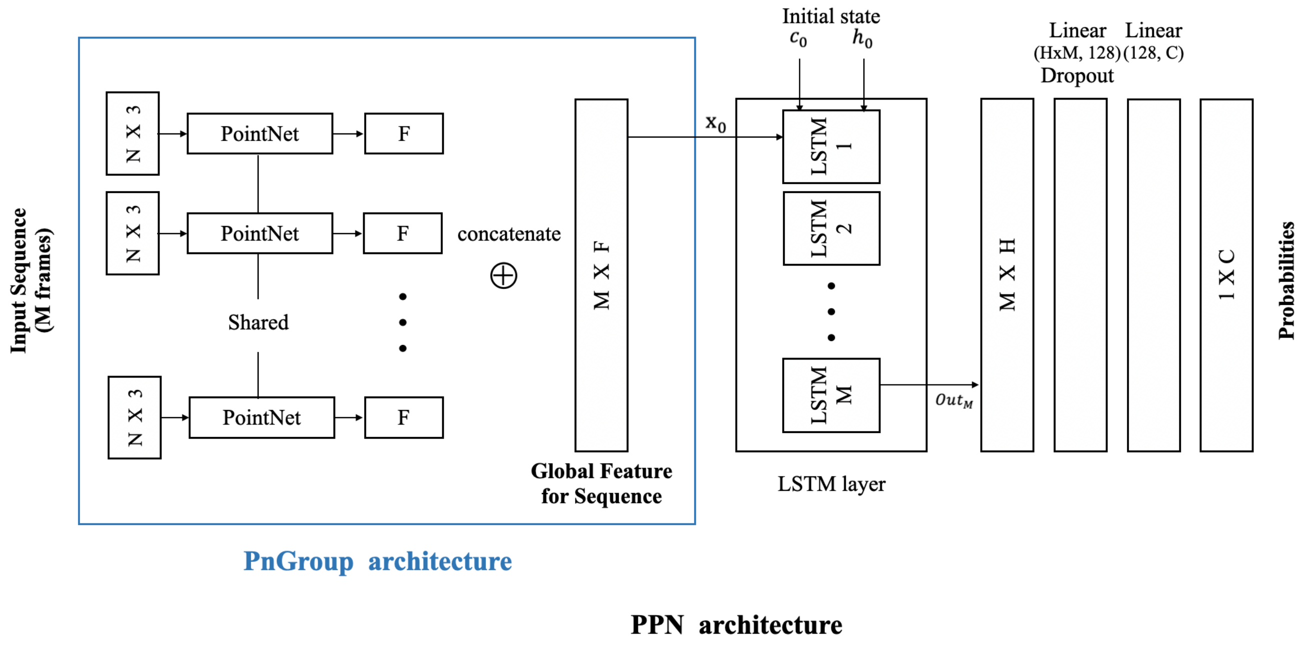

Figure 5. The architecture of our proposed model. The model takes input sequence as an input, and outputs a probability distribution over classes. M refers to the number of frames, N refers to the number of points, F is the number of output features, H is the hidden size of LSTM, and C is the number of classes.

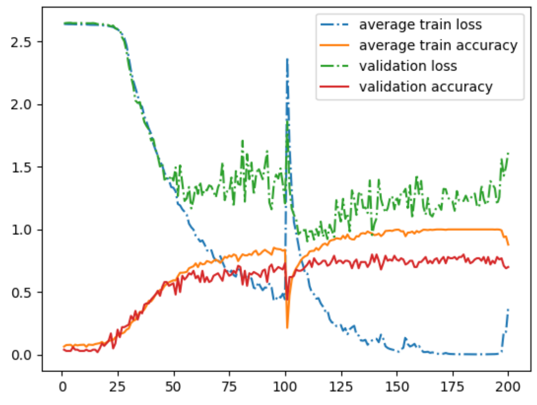

Figure 6. wd = $1e^{-5}$.

fore it learn less, let alone the fact that we never really trained our feature extraction module. We tried implementing an autoencoder and use the code as our feature to feed to the LSTM module. However, because of some technical issues with our machine on the GPU memory we cannot achieve that. But we believe this unsupervised learning approach to tackle the lack of joint location data would be perfect and should perform quite close to the approaches with the joint location data.

References

[1] Yuxiao Chen, Long Zhao, Xi Peng, Jianbo Yuan, and Dimitris N. Metaxas. Construct dynamic graphs for hand gesture recognition via spatial-temporal attention. *CoRR*, abs/1907.08871, 2019. 1

[2] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeborre. Skeleton-based dynamic hand gesture recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1206–1214, 2016. 1

[3] Guillaume Devineau, Fabien Moutarde, Wang Xi, and Jie Yang. Deep learning for hand gesture recognition on skeletal data. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 106–113. IEEE, 2018. 1, 4

[4] Y. Eldar, M. Lindenbaum, M. Porat, and Y.Y. Zeevi. The farthest point strategy for progressive image sampling. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 2 - Conference B: Computer Vision Image Processing. (Cat. No.94CH3440-5)*, pages 93–97 vol.3, 1994. 2

[5] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns, 2016. 4

[6] Kenneth Lai and Svetlana N. Yanushkevich. CNN+RNN depth and skeleton based dynamic hand gesture recognition. *CoRR*, abs/2007.11983, 2020. 1

[7] Chaoyu Liang, Yonghong Song, and Yuanlin Zhang. Hand gesture recognition using view projection from point cloud. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4413–4417, 2016. 1

[8] Hasan Mahmud, Mashrur Mahmud Morshed, and Md. Kamrul Hasan. A deep-learning-based multimodal depth-aware dynamic hand gesture recognition system. *CoRR*, abs/2107.02543, 2021. 1

[9] Yuecong Min, Yanxiao Zhang, Xiujuan Chai, and Xilin Chen. An efficient pointlstm for point clouds based gesture recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5760–5769, 2020. 1

[10] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. 1, 3

[11] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. 1

[12] Dariush Salami, Sameera Palipana, Manila Kodali, and Stephan Sigg. Motion pattern recognition in 4d point clouds. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2020. 1, 3

[13] Quentin De Smedt, Hazem Wannous, Jean-Philippe Vandeborre, J. Guerry, B. Le Saux, and D. Filliat. 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. In Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2017. 2