Hand PointNet: 3D Hand Pose Estimation using Point Sets Supplementary Material

Liuhao Ge[1], Yujun Cai[1], Junwu Weng[2], Junsong Yuan[3]
[1]Institute for Media Innovation, Interdisciplinary Graduate School, Nanyang Technological University
[2]School of Electrical and Electronic Engineering, Nanyang Technological University
[3]Department of Computer Science and Engineering, State University of New York at Buffalo
{ge0001ao, yujun001, we0001wu}@e.ntu.edu.sg, jsyuan@buffalo.edu

1. Network Architecture of Hand Pose Regression Network

We present the detailed network architecture of the hand pose regression network in Figure 1, which is based on the architecture of hierarchical PointNet proposed in [7]. As can be seen, the hand pose regression network has three set abstraction levels. At the l-th level $(l = 1, 2)$, N_l points are sampled using iterative farthest point sampling and k nearest neighboring points of each sampled point are grouped as a local region. Following [7], we search nearest points that are within a radius to the query point in the Euclidean space. After the process of sampling and grouping, we get N_l local regions. Each local region containing k points is fed into a basic PointNet [6] to extract a C_l-dim local feature. The output data size of this level is $N_l \times C_l$. The C_l-dim local features together with the d-dim coordinates of the N_l sampled points are input to the next level. At the 3^{rd} level, all the input points of this level are fed into a basic PointNet to extract a C_3-dim global feature. The global feature is mapped to an F-dim output vector through a multi-layer perceptron (MLP) network.

Each MLP network in Figure 1 is composed of several fully connected layers. All fully connected layers are followed by batch normalization and ReLU except for the last layer of the last MLP network. We do not use dropout layer in our implementation.

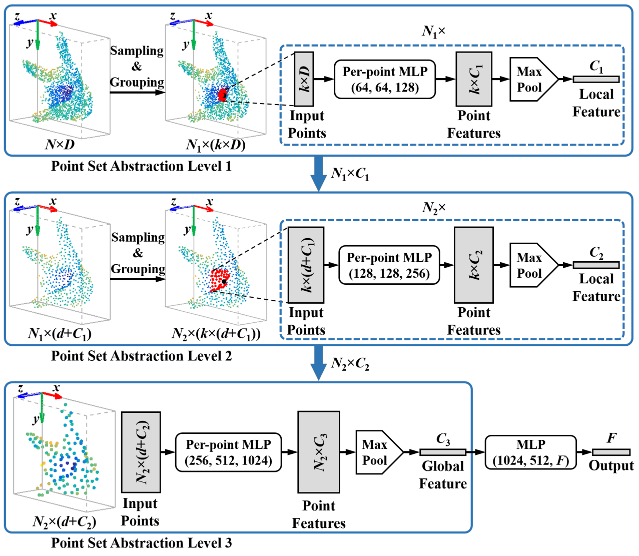

Figure 1: Network architecture of the hand pose regression network which is based on the hierarchical PointNet [7]. Numbers in parentheses of MLP networks are layer sizes. In our implementation, $N = 1024$, $N_1 = 512$, $N_2 = 128$, $k = 64$, $D = 6$, $d = 3$, $C_1 = 128$, $C_2 = 256$, $C_3 = 1024$. We set the dimension of output vector as $F = 2 \times M < 3 \times M$, where M is the number of hand joints, $3 \times M$ is the dimension of hand joint locations.

2. Additional Experiments

2.1. Impact of Surface Normal

We evaluate the impact of surface normal on estimation errors. As shown in Figure 2, using surface normal as the input feature will improve the performance a little bit. We also experiment with approximating surface normal using different numbers (k) of nearest neighboring points. As can be seen, our method is not sensitive to the procedure of surface normal approximation.

2.2. Impact of Fingertip Refinement

As shown in Table 1, the improvement of fingertip refinement on overall mean error is small (0.1mm$\sim$0.3mm), since the five fingertips only take a small part of all the joints. Moreover, without the fingertip refinement, our method can still achieve state-of-the-art performance on all the three datasets.

In addition, we use the estimation results of other methods as initial estimations for fingertip refinement. As shown

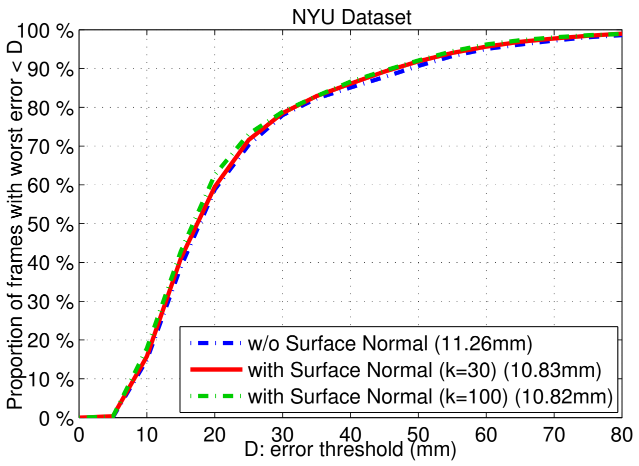

Figure 2: The impact of surface normal on the proportion of good frames (NYU [10] dataset). Mean errors are shown in parentheses.

Table 1: The impact of fingertip refinement on NYU [10], MSRA [8] and ICVL [9] datasets.

	Mean Error on **All Joints**		Mean Error on **5 Fingertips**	
	w/o Refine.	with Refine.	w/o Refine.	with Refine.
NYU [10]	10.8mm	10.5mm	13.9mm	13.1mm
MSRA [8]	8.6mm	8.5mm	11.4mm	10.7mm
ICVL [9]	7.2mm	6.9mm	8.9mm	8.0mm

Table 2: The impact of our fingertip refinement using estimation results of other methods as initial estimations (NYU [10] dataset).

	Mean Error on **All Joints**		Mean Error on **5 Fingertips**	
	w/o Refine.	with Refine.	w/o Refine.	with Refine.
3D CNN [2]	14.1mm	13.5mm	18.7mm	16.9mm
DeepPrior++ [4]	12.3mm	11.8mm	16.1mm	14.7mm
Pose-REN [1]	11.8mm	11.6mm	14.7mm	14.1mm

in Table 2, our fingertip refinement method can also improve fingertip estimations of other methods evidently, and slightly improve the mean error on all joints.

2.3. Additional Comparison with State-of-the-arts

In order to make a fair comparison with the spatial attention network-based hierarchical hybrid method proposed in [11], we evaluate the proportion of hand joints within different error thresholds on NYU hand pose dataset [10]. Following the experimental setting in [11], we calculate this evaluation metric on 11 hand joints (removing the palm joints except the root joint of thumb in the set of original 14 hand joints on NYU dataset).

Experimental results are shown in Figure 3. Apart from the spatial attention network-based method [11], we also compare our method with DeepPrior [5], 3D CNN [2], REN [3] and DeepPrior++ [4] methods by using this evalu-

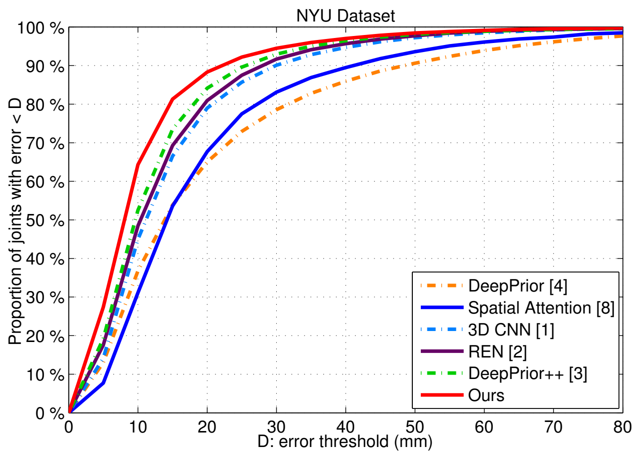

Figure 3: Comparison with state-of-the-art methods [5, 11, 2, 3, 4] on NYU [10] dataset. The proportions of joints within different error thresholds are presented in this figure.

ation metric. As can be seen, our method is superior to all of these methods over all the error thresholds. For example, when the error threshold is 10mm, the proportion of joints within this error threshold of our method is about 30% more than that of the spatial attention network-based method [11], 20% more than that of the 3D CNN-based method [2] and 10% more than that of the DeepPrior++ method [4].

3. More Details on Runtime and Model Size

During the testing stage, the depth image is first converted to a set of 3D points, from which 1024 points are randomly sampled. Since we use the hierarchical PointNet [7], the farthest point sampling (FPS) algorithm is applied to evenly sample 512 and 256 points from the 1024 points for the first two set abstraction levels. This step including random sampling and FPS takes 1.7ms for one frame in average. The next step is surface normal approximation for the 1024 points, which takes 6.5ms for one frame in average. These two steps are implemented using C++ with the Point Cloud Library (PCL). The 3D points attached with surface normals are then fed into the hand pose regression network to estimate 3D hand joint locations, which takes 9.2ms for one frame in average on a single Nvidia GTX1080 GPU with a batch size 1. For fingertip refinement, neighboring points search and surface normal approximation take 2.8ms for one frame in average, and the fingertip refinement network forward propagation takes 0.3ms for one frame in average on a single Nvidia GTX1080 GPU. The batch size is equal to the number of straightened fingers for each frame. Therefore, the total runtime of our method is 20.5ms for one test frame in average, and the average frame rate is about 48fps.

The number of parameters in the hand pose regression network is about 2.3×10^6 (2.3M). These parameters are stored in 32 bit float and the size of parameters is 9.2MB. The number of parameters in the fingertip refinement network is about 1.4×10^5 (142K). These parameters are stored in 64 bit double and the size of parameters is 1.1MB. In total, there are about 2.5×10^6 (2.5M) parameters in these two networks and the total size of network parameters is 10.3MB.

References

[1] X. Chen, G. Wang, H. Guo, and C. Zhang. Pose guided structured region ensemble network for cascaded hand pose estimation. *arXiv preprint arXiv:1708.03416*, 2017.

[2] L. Ge, H. Liang, J. Yuan, and D. Thalmann. 3D convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *CVPR*, 2017.

[3] H. Guo, G. Wang, X. Chen, C. Zhang, F. Qiao, and H. Yang. Region ensemble network: Improving convolutional network for hand pose estimation. In *ICIP*, 2017.

[4] M. Oberweger and V. Lepetit. Deepprior++: Improving fast and accurate 3d hand pose estimation. In *ICCV Workshop*, 2017.

[5] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands deep in deep learning for hand pose estimation. In *CVWW*, 2015.

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017.

[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.

[8] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun. Cascaded hand pose regression. In *CVPR*, 2015.

[9] D. Tang, H. J. Chang, A. Tejani, and T. K. Kim. Latent regression forest: Structured estimation of 3D articulated hand posture. In *CVPR*, 2014.

[10] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics*, 33(5):169, 2014.

[11] Q. Ye, S. Yuan, and T.-K. Kim. Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation. In *ECCV*, 2016.