Point-to-Pose Voting based Hand Pose Estimation using Residual Permutation Equivariant Layer

Shile Li
Technical University of Munich
li.shile@mytum.de

Dongheui Lee
Technical University of Munich, German Aerospace Center
dhlee@tum.de

Abstract

Recently, 3D input data based hand pose estimation methods have shown state-of-the-art performance, because 3D data capture more spatial information than the depth image. Whereas 3D voxel-based methods need a large amount of memory, PointNet based methods need tedious preprocessing steps such as K-nearest neighbour search for each point. In this paper, we present a novel deep learning hand pose estimation method for an unordered point cloud. Our method takes 1024 3D points as input and does not require additional information. We use Permutation Equivariant Layer (PEL) as the basic element, where a residual network version of PEL is proposed for the hand pose estimation task. Furthermore, we propose a voting-based scheme to merge information from individual points to the final pose output. In addition to the pose estimation task, the voting-based scheme can also provide point cloud segmentation result without ground-truth for segmentation. We evaluate our method on both NYU dataset and the Hands2017Challenge dataset, where our method outperforms recent state-of-the-art methods.

1. Introduction

Hand pose estimation plays an important role in human-robot interaction tasks, such as gesture recognition and learning grasping capability by human demonstration. Since the emergence of consumer level depth sensing devices, a lot of depth image based hand pose estimation methods appeared. Many state-of-the-art methods use depth image as input, which provides the conveniences to use the well developed convolutional neural networks or residual networks. However, methods using 2D images as input cannot fully utilize 3D spatial information in the depth image. Furthermore, the appearance of the depth image is dependent on the camera parameters, such that the trained model using one camera's image cannot generalize well to another camera's image. On the other hand, 3D data is more "direct" and "distinctive" than depth image because the appearance of 3D data is unique and invariant to camera parameters.

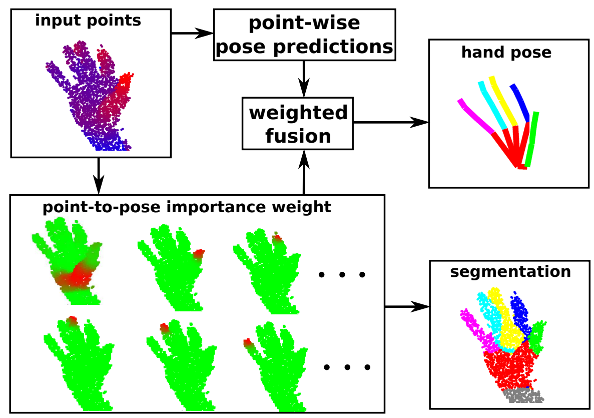

Figure 1. Our method takes point cloud as input. Then each point predicts the hand pose and its importance weights for different pose dimensions. The final pose is obtained through weighted fusion from each point's pose prediction. Using the importance weight, the hand can be clearly segmented into different parts, although no segmentation ground-truth was used during training.

Recently, methods using 3D data as input have shown the outperformance over depth image based methods [36]. One way to use 3D input data is to convert 2D depth image to volumetric representation, such as 3D voxels [15] [4], where occupied 3D voxel is set to 1 and voxels with empty space is set to 0. Using the voxelized data brings the convenience to directly use 3-dimensional CNN learning structure. However, the voxelization requires large amount of memory to represent the input and output data, which prevents the deployment of a very deep structure.

Another way to use 3D input data is to use unordered point cloud as input [6][9][3]. Recently, PointNet, a deep learning structure for point cloud, has shown its success in different tasks. The PointNet estimates point-wise features for individual points and extract global feature from individual points using a max-pooling layer, such that the network is invariant to the order of points. Ge et al. use Point-

Net [20][22] as backbone to estimate hand pose from point cloud [6]. However, tedious pre-processing steps such as surface normal estimation and k-nearest-neighbours search are required for [6]. Moreover, the final max-pooling layer in the PointNet neglects many informations that might be crucial for pose estimation.

In this work, we explore a more flexible learning structure for unordered point sets, the permutation equivariant layer (PEL) [24] [39]. The PEL is a deep learning structure that can be applied for unordered points. In PEL, point-wise features are computed, where each point's feature does not only depend on its own input, but also the global maximum value. Using PEL as the basic element, we propose a residual network version of PEL to construct a deep network for hand pose estimation task. Moreover, we propose a point-to-pose voting scheme to obtain hand pose, which eliminates the use of max-pooling layer to extract global feature, thus avoiding the loss of information. Furthermore, the generated point-to-pose importance weights can be also used for the hand segmentation task (Fig. 1), where clear segmentation result can be obtained even without the segmentation ground-truth.

The contributions of this work are:

- We propose a novel deep learning based hand pose estimation method for unordered point cloud. Using Permuation Equivariant Layer as the basic element, a residual network version of PEL is used to solve the hand pose estimation task. Compared to PointNet [22] based methods, our method doesn't require tedious steps such as normal estimation, nearest neighbour estimation.

- We propose a point-to-pose voting scheme to merge the information from point-wise local features, which also genearates weakly-supervised segmenation results without the need of segmentation ground-truth.

- We evaluate our method on Hands2017 Challenge dataset and NYU dataset, where state-of-the-art performance is shown. The proposed method achieves the lowest pose error on the Hands2017 Challenge dataset at the time of submission.

2. Related work

A lot of research about hand pose estimation has been done in the last decade, which can be categorized to generative, discriminative and hybrid methods. Generative methods rely on a hand model and an optimization method to fit the hand model to the observations [25][29][23][19]. Discriminative methods use learning data to learn a mapping between observation and the hand pose [17][30][15][4][3][16][26][28]. Hybrid methods use a combination of the generative and discriminative methods [18][27][34]. Our method is a learning based method thus falls into the second category.

Deep learning for hand pose estimation

With the success of deep learning methods for 2D computer vision, depth image based deep learning methods also showed good performance in hand pose estimation task. Tompson et al. use 2D CNN to predict heatmaps of each joint and then rely on PSO optimization to estimate the hand pose [30]. Oberweger et al. [17] uses 2D CNN to directly regress the hand pose out of the image features, where a bottleneck layer was used to force the predicted pose obey certain prior distribution. In a later work, Oberweger [16] replaced CNN to a more sophisticated learning structure, ResidualNet50, to improve the performance of feature extraction. Zhou et al. [40] regress a set of hand joint angles and feed the joint angles into an embedded kinematic layer to obtain the final pose. Ye et al. [33] use a hierarchical mixture density network to handle the multi-modal distribution of occluded hand joints.

Recently, 3D deep learning has been also applied for the hand pose estimation task. Moon et al. use 88^3 voxels to represent hand's 3D geometry and use 3D CNN to estimate hand pose [15]. Their method achieved very accurate result, however, 3D voxelization of the input and output data requires large memory size, such that their method only runs at 3.5 FPS. Ge et al. [6][9] use 1024 3D points as input, and rely on PointNet [22] structure to regress the hand pose. Their method achieved satisfying performance, but tedious pre-processing steps are required, which includes oriented bounding box (OBB) calculation, surface normal estimation and k-nearest-neighbours search for all points. Chen et al. improves Ge's method by using a spatial transformer network to replace the OBB and furthermore added a auxiliary hand segmentation task to improve the performance [3]. Their method can be trained end-to-end without OBB, but the segmentation ground-truth data require a extra precomputation step from the pose data.

3D Deep learning

Since 3D data cannot be directly fed into a conventional 2D CNN, some methods project the 3D data onto different views to obtain multiple depth images and perform CNN on all images [7] [21] [10] [35]. Another way to process the 3D data is to use volumetric representation and process the data with 3D CNN [8] [32] [14] [15]. These methods can capture the feature of input data more effective, but they require large memory size. Qi et al. developed PointNet to handle unordered point cloud [20]. The PointNet estimates point-wise local features and obtains global features with a max-pooling layer. Later on, PointNet++ extends PointNet by hierarchically upsampling the local features into higher levels [22].

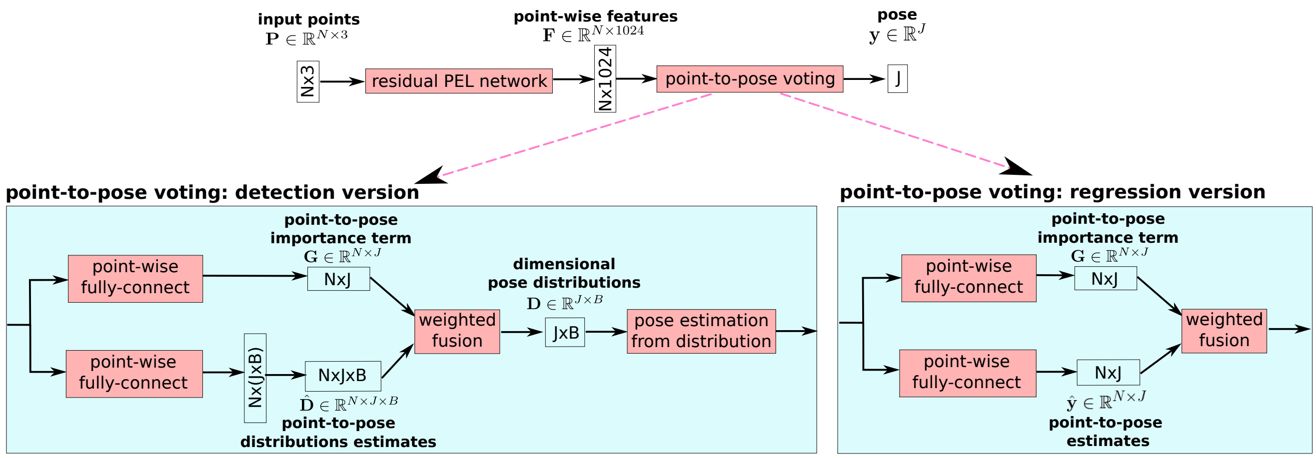

Figure 2. Overview of our method.

Other recent methods taking 3D points as input include point-wise CNN [11], Deep kd-Networks [12], Self-Organizing Net [13] and Dynamic Graph CNN [31]. Despite their good performance for different tasks, they all require extra steps to estimate k-nearest neighbours or construct kd-tree, which are not required in our proposed residual PEL network.

3. Methodology

The overview of our method is illustrated in Fig. 2. Our method takes N 3D points $\mathbf{P} \in \mathbb{R}^{N\times 3}$ with arbitrary order as input, and outputs the vectorized 3D hand pose $\mathbf{y} \in \mathbb{R}^{J}$ in the end, where $J = 3 \times \#joints$. To estimate the hand pose, the residual permutation equivariant layers (PEL) (Fig. 4) first extract features from each point (Section 3.2). Using the point-wise local features, we use point-to-pose voting to estimate the final pose output (Section 3.3), where two versions for point-to-pose voting are developed, which are the detection version and the regression version.

3.1. Pre-processing with view normalization

For pre-processing, first, the depth pixels in the hand region are converted to 3D points. The next step is to create a 3D bounding box for the hand points to obtain normalized coordinate of these points. A usual pre-processing method will simply create a bounding box aligned with the camera coordinate system (Fig 3a). However, because of self-occlusion of the hand, this will result in different set of observation points for the exact same pose label, which creates one-to-many mapping of the input-output pairs.

To maintain the one-to-one mapping relation of the input-output pairs, we propose to use view normalization to align the bounding box's z-axis $[0,0,1]^T$ with the view direction towards the hand centroid point $\mathbf{c} \in \mathbb{R}^3$. The alignment is performed by rotating the hand points with a rotation matrix $\mathbf{R}_{cam}$:

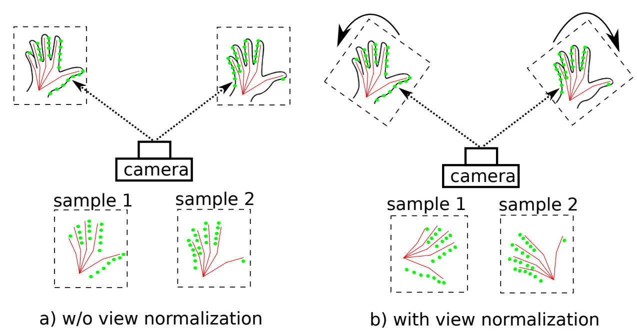

Figure 3. View normalization as pre-processing step. Red skeletons indicate ground-truth pose, green points indicate observed points of the camera. a) The same hand pose result in different observations due to different view directions, thus the resulted training samples will contain one-to-many mappings. b) With view normalization, the different observations will also have different pose labels, thus the input-output pairs will have a one-to-one mapping.

$$\begin{aligned} \alpha_y &= atan2(\mathbf{c}_x, \mathbf{c}_z), \\ \widetilde{\mathbf{c}} &= \mathbf{R}_y(-\alpha_y) \cdot \mathbf{c}, \\ \alpha_x &= atan2(\widetilde{\mathbf{c}}_y, \widetilde{\mathbf{c}}_z), \\ \mathbf{R}_{cam} &= \mathbf{R}_y(-\alpha_y) \cdot \boldsymbol{R}_x(\alpha_x). \end{aligned} \tag{1}$$

After rotating the observation points and ground truth pose with $\mathbf{R}_{cam}$, the hand is rotated such that it appears right in front of the camera, As illustrated in Fig. 3b, the one-to-many mapping problem is then avoided.

3.2. Residual Permutation Equivariant Layers

The feature extraction module in our method is called Residual Permutation Equivariant Layers. The basic element is the permutation equviariant layer (PEL), which follows the design from [24]. A PEL takes a set of unordered points as input and computes separate features for each individual input point.

Assuming that the input for a PEL is $\mathbf{x} \in \mathbb{R}^{N \times K_{in}}$ and the output is $\mathbf{x}' \in \mathbb{R}^{N \times K_{out}}$, where N is the number of points and K_{in}, K_{out} are the size of input and output feature dimensions. The output $\mathbf{x}'$ of the PEL is:

$$\mathbf{x}' = \sigma(\mathbf{1}_N \boldsymbol{\beta}^T + (\mathbf{x}\,\mathrm{diag}(\boldsymbol{\lambda}) + \mathbf{1}_N \mathbf{x}_{max}^T\,\mathrm{diag}(\boldsymbol{\gamma}))\mathbf{W}), \quad (2)$$

where $\lambda \in \mathbb{R}^{K_{in}}$,$\gamma \in \mathbb{R}^{K_{in}}$ are weighting terms for the point's own feature and the global maximum value respectively, and $\mathbf{x}_{max} \in \mathbb{R}^{K_{in}}$ is a vector representing maximum values for each column of $\mathbf{x}$. $\mathbf{W} \in \mathbb{R}^{K_{in} \times K_{out}}$ is the weight term and $\boldsymbol{\beta} \in \mathbb{R}^{K_{out}}$ is the bias term and $\mathbf{1}_N \in \mathbb{R}^N$ is a vector full of ones. Furthermore, an activation function $\sigma(\cdot)$ is applied to provide non-linearity, where a sigmoid function is used in our method.

This layer is invariant to input order because the output value of each individual point only depends on its own input feature and the global maximum values in each feature dimension, whereas the global maximum values are also invariant to the order of input points.[1] In this way, each point's feature does not only computed based on its own input feature, each point also exchanges information with other points through the weighted summation of $\mathbf{x}_{max}$.

For the practical side, four elements need to be trained, which are β, $\boldsymbol{\lambda}$, $\boldsymbol{\gamma}$ and $\mathbf{W}$. In total, the number of parameters needed for one layer is $K_{out} + (K_{out} + 2)K_{in}$, which is only slightly more than a fully-connected layer, thus it is feasible for training in practice.

In order to extract very complex features, we construct a residual network with 39 PEL layers. As illustrated in Fig. 4, we use three residual blocks, whereas each residual block consists of 13 PELs and four short-cut connections. Furthermore, after each PEL, a batch normalization is performed.

3.3. Point-to-pose voting

With the residual PEL module, features $\mathbf{F}$ of points are computed, where each row of $\mathbf{F}$ represents local feature for one point. Using these local point-wise features, the hand pose $\mathbf{y} \in \mathbb{R}^J$ will be estimated using a point-to-pose voting scheme. Two versions for point-to-pose voting are explored, which are the detection based version and the regression based version. The performance of these two versions will be compared in the experiment section.

Detection version

In the detection version (Fig. 2 left), probability distributions of each pose dimension is firstly detected and the pose is then integrated from the distributions. We use two

[1]The detailed proof of the invariance for PEL can be found in [24].

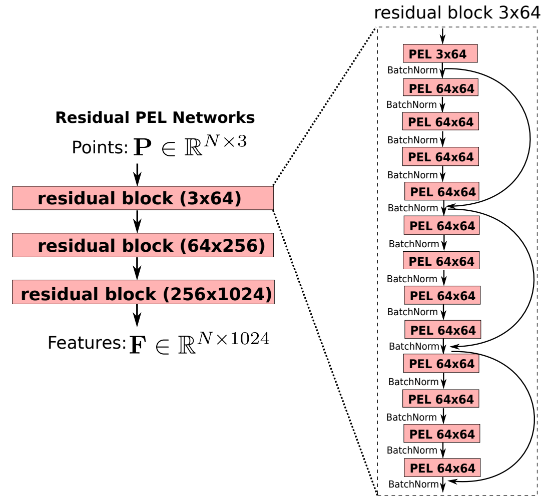

Figure 4. Residual network of permutation equivariant layer

separate fully connected modules to estimate two matrices: an importance term $\mathbf{G} \in \mathbb{R}^{N \times J}$ and a distributions $\hat{\mathbf{D}} \in \mathbb{R}^{N \times J \times B}$. An element of importance matrix $\mathbf{G}_{nj}$ represents the confidence level of nth input point to predict the jth output pose dimension. In other words, each of the N points predicts J B-dimensional distributions and J corresponding importance weights. Notice that the final layer of the two fully connected modules are sigmoid functions, such that all elements of $\mathbf{G}$ and $\hat{\mathbf{D}}$ are in the range of $[0, 1]$.

$\hat{\mathbf{D}}$ represents the output pose distributions, where each point makes its own predictions to J output dimensions. Each of the output pose dimension is represented as discrete distribution using a B bins, representing the value range in $[-r, +r]$ with the resolution per bin $\Delta d = 2r/B$. For the jth dimension of the output pose $\mathbf{y}_j$, the corresponding bin index for itself is then:

$$index_j^{gt} = \lceil (\mathbf{y}_j^{gt} + r)/\Delta d \rceil,$$

and the ground truth distribution is defined as:

$$\mathbf{D}_{jb}^{gt} = \begin{cases} 1, & \text{if } b \in [index_j^{gt} - 1, index_j^{gt} + 1] \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

whereas the three bins around ground truth pose are set to one and all other bins are set to zero.

The final distribution for the J-dimensional output $\mathbf{D} \in \mathbb{R}^{J \times B}$ is then obtained by merging the predictions of all N points:

$$\mathbf{D}_{jb} = \frac{\sum_{n=1}^{N} (\mathbf{G}_{nj} \hat{\mathbf{D}}_{njb})}{\sum_{n=1}^{N} \mathbf{G}_{nj}}. \quad (4)$$

And the final pose $\mathbf{y}$ is estimated with integration over the distribution:

$$\mathbf{y}_j = \frac{\sum_{b=1}^{B} (b - 0.5)\mathbf{D}_{jb}}{\sum_{b=1}^{B} \mathbf{D}_{jb}}, \quad (5)$$

where $b-0.5$ represents the bin center position.

Regression version

In the regression version (Fig. 2 right), each point will directly predict the pose without the intermediate distribution detection. Similarly to the detection version, two separate fully connected modules are used to estimate the importance term $\mathbf{G} \in \mathbb{R}^{N\times J}$ and the point-to-pose estimates $\hat{\mathbf{y}} \in \mathbb{R}^{N\times J}$. Then the final pose output is merged as the weighted average over all points' predictions:

$$\mathbf{y}_j = \frac{\sum_{n=1}^{N}(\mathbf{G}_{nj}\hat{\mathbf{y}}_{nj})}{\sum_{n=1}^{N}\mathbf{G}_{nj}}. \tag{6}$$

3.4. Segmentation using importance term

The importance term $\mathbf{G} \in \mathbb{R}^{N\times J}$ is estimated automatically without the ground-truth information. However, it still provides vital information of each point's importance to the pose output. Therefore, the obtained importance term can be also used for the hand segmentation task based on the most contributed pose dimension. For the n-th point having the importance terms $\mathbf{g} = \mathbf{G}_n$, the point's most contributed pose dimension is:

$$j_{max} = \operatorname*{argmax}_j \mathbf{g}_j,$$

where the pose dimension j_{max} can be categorized to a specific hand part. In this work, we categorized the J pose dimensions to palm, thumb, index, ring and pinky fingers.

3.5. Training Loss

The only training loss for the detection version is the logarithm loss of the pose distributions:

$$\begin{aligned} L_{det} = -\sum_{j=1}^{J}\sum_{b=1}^{B} &\mathbf{D}_{jb}^{gt} log(\mathbf{D}_{jb}+\epsilon) \\ &+(1-\mathbf{D}_{jb}^{gt})(1-log(\mathbf{D}_{jb}+\epsilon)), \end{aligned} \tag{7}$$

where $\epsilon = 10^{-7}$ is a small offset to avoid feeding zero to the logarithm operator.

The only training loss used for the regression version is the L2 loss between predicted pose and ground-truth pose:

$$L_{reg} = \frac{1}{2}\sum_{j=1}^{J}(\mathbf{y}_j^{gt} - \mathbf{y}_j)^2. \tag{8}$$

For both detection and regression versions, the importance term $\mathbf{G} \in \mathbb{R}^{N\times J}$ is estimated automatically without the ground-truth information.

4. Experiment and result

Our hand pose estimation method is evaluated on the Hands2017Challenge dataset [37] and the NYU [30] dataset. The Hands2017Challenge is composed from parts of the Big Hand 2.2M dataset [38] and the First-person Hand Action Dataset (FHAD) [5], it is currently the largest dataset available. Its training set contains 957032 depth images of five different hands. The test set consists of 295510 depth images of ten different hand shapes, of which five are the same as in the training set and five are entirely new. The NYU dataset contains 72757 training images of a single subject's hand and 8252 test images that include a second hand shape besides the one from the training set. The NYU dataset provides depth images from three different views, we trained our method both using only frontal view data and using all three views. And we test only using the frontal view.

Our method is implemented using TensorFlow [1]. The networks are trained on a PC with an AMD FX-4300/Intel Core i7-860 CPU and an nVidia GeForce GTX1060 6GB GPU. We train 100 epochs for the NYU dataset and train only 20 epochs for the Hands 2017 challenge dataset since the challenge dataset has a large size. For both datasets, the first 50% of the epochs are trained with smaller number of points ($N = 256$) to boost the training speed. The remaining epochs are trained with a point size of $N = 512$. We used Adam optimizer for training with an initial learning rate of 10^{-3} and we decrease the learning rate to 10^{-4} for the last 10% of the epochs. For the detection version, we set $r = 15mm$ and $B = 60$. Online augmentation was performed with random translation in all three dimensions within $[-15, 15]mm$, random scaling within $[0.85, 1.15]$ and random rotation around z-axis within $[-\pi, \pi]$.

4.1. Evaluation metrics

For the NYU dataset, two standard metrics are used to evaluated the performance. The first metric is the mean joint error, which measures the average Euclidean distance error for all joints across the whole test set. The second metric is correct frame proportion, which indicates the proportion of frames that have all joints within a certain distance to ground truth. The second metric is considered as more difficult since single joint violation will cause an unqualified frame. For the Hands2017Challenge dataset, only the mean joint error is used since the ground-truth data of test set is not publicly available and the official test website only provides the mean joint error result.

4.2. Self-comparison

In this subsection, we perform self-comparison to show the effects of different components in our method. The detailed comparison can be found in Table 1.

	Hands2017Challenge dataset			NYU dataset		
	detection	detection w/o view normalization	regression	detection/ single view	regression/ single view	regression/ three views
256 points	11.34	13.14	11.21	9.82	9.45	9.05
512 points	10.23	11.93	10.11	9.33	9.06	8.49
1024 points	9.93	11.67	9.82	9.25	8.99	8.35
2048 points	9.93	11.69	9.87	9.32	9.08	8.35

Table 1. Self-comparison result

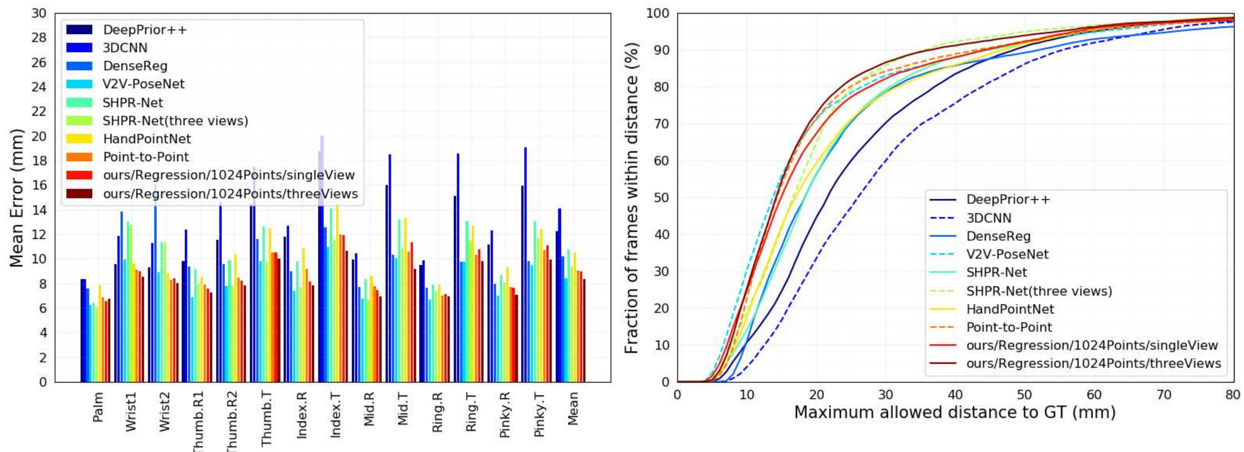

Figure 5. Comparison with state-of-the-arts on NYU [30] dataset. Left: mean errors of different joints. Right: proportion of correct frames based on different error thresholds.

method	**avg test**	**seen test**	**unseen test**
Ours/regression	9.82	7.15	12.04
Ours/detection	9.93	7.18	12.22
V2V-PoseNet [15]	9.95	6.97	12.43
RCN-3D [36]	9.97	7.55	12.00
oasis [6]	11.30	8.86	13.33
THU VCLab [2]	11.70	9.15	13.83
Vanora [36]	11.91	9.55	13.89

Table 2. Comparison of our method with state-of-the-art methods on the Hands2017Challenge dataset

method	**mean joint error (mm)**
Ours/regression/singleView	8.99
Ours/regression/threeViews	8.35
Ours/detection	9.25
DeepPrior++ [16]	12.23
3DCNN	14.11
DenseReg [4]	10.21
V2V-PoseNet [15]	8.42
SHPR-Net [3]	10.77
SHPR-Net (three views) [3]	9.37
HandPointNet [6]	10.54
Point-to-Point [9]	9.04

Table 3. Comparison of our method with state-of-the-art methods on the NYU dataset

View normalization. To validate the necessity of view normalization, we trained our method using both view normalized data and original data for the detection version. It is evident from Table 1 that view normalization decreases the pose estimation error by about 1.5 mm for the Hands2017Challenge dataset.

Detection vs. regression. Yuan et.al. indicates that detection based methods work in general better than regression-based methods [36], therefore we implemented both detection-based (ours/distribution) and regression-based (ours/regression) variations. As seen from Table 1, in both datasets, both variations show similar performance, where regression-based variation slightly outperforms the detection-based counterpart. Possible reasons for this can be quantization effect of the binary distribution and the simplification of 1-dimensional heat vectors compared to 2D or 3D heat maps used in previous works. However, the 1D heat vector representation is much more efficient than the 3D heatmap representation. For the heat vectors, we need $B \times J$ values to represent the pose output, whereas 3D heatmaps require $J \times B^3$ values [15]. In future work, it is worth to investigate more different loss types and heat map representations.

Number of points. Taking advantage of the PEL structure and voting-based scheme, our method is very flexible to the input point cloud size. Although the network was trained with 512 points, arbitrary number of points can be used at the testing stage. For an online application, this property can be beneficial to choose an arbitrary number of points based on the computational resources available. As seen from Table 1, different number of points were tested

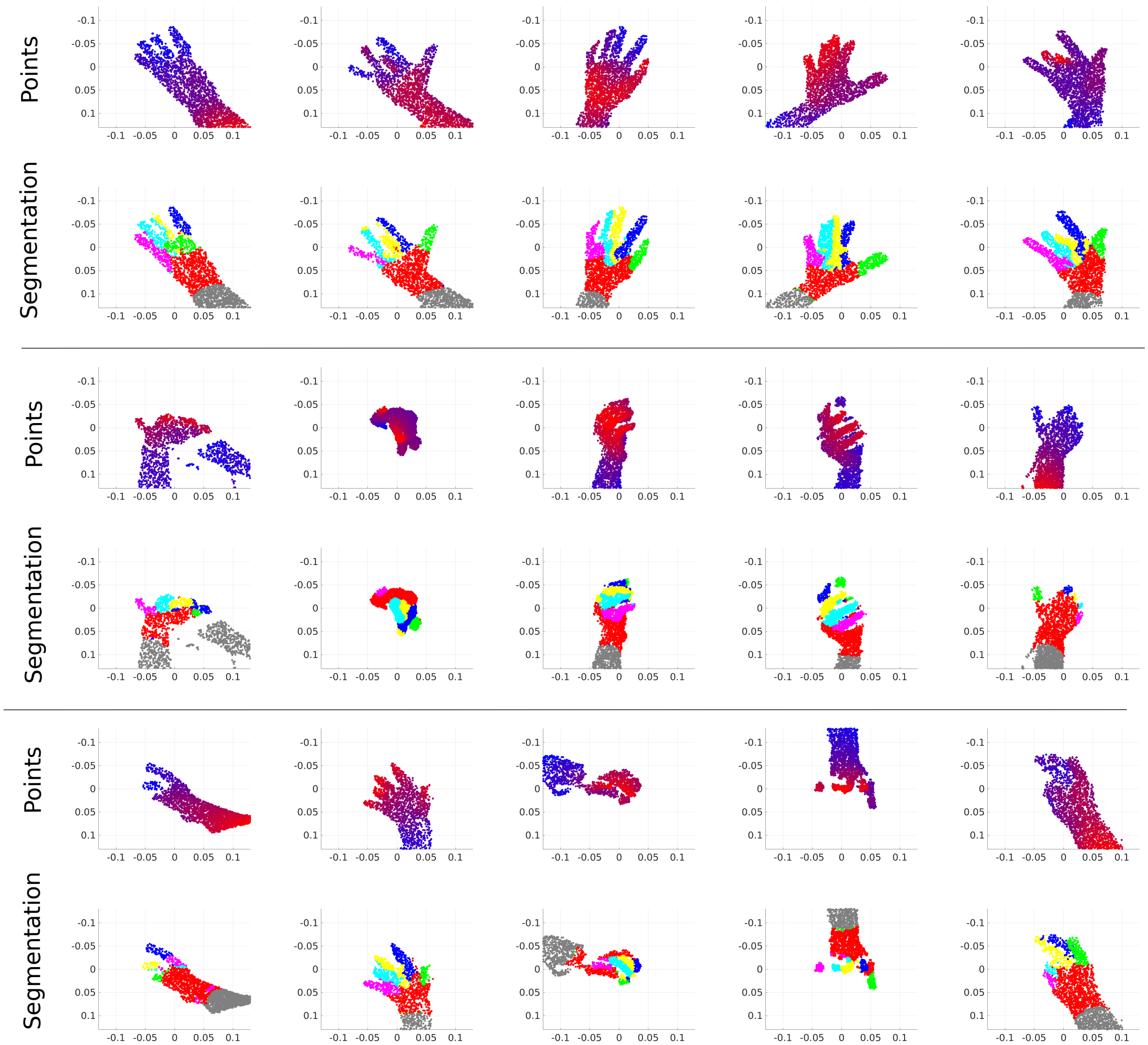

Figure 6. Segmentation results based on importance weights (best viewed in color). Points: input point cloud, color indicates depth value, blue points are more distanced and red points are more closer to the camera. Segmentation: each part of the hand is indicated with an different color, palm (red), thumb (green), index (blue), middle (yellow), ring (cyan), pinky (pink) and irrelevant points with low importance weight for all parts (gray).

for both datasets. Our method can achieve good performance with only 256 points, the mean joint error only increased by 0.11 mm compared to 512 points. In general, more points provides better performance, but it doesn't improve any more after 1024 points. Therefore, we choose 1024 points for testing to compare our method with other state-of-the-art methods.

4.3. Comparison to state-of-the-art methods

Hands2017Challenge dataset. Since the ground-truth data for the testing set publicly available, some previous papers divide the training set on their own to create their own testing set. Therefore, for fair comparison, we only compare to those methods, who have also tested on the official testing website[2]. In Table 2, we compare our method with five other top performing methods on the Hands2017Challenge dataset, which include both methods using 3D input data and methods using 2D depth image. RCN-3D [36], THU VCLab [2] and Vanora [36] use depth image as input data. V2V-PoseNet [15] uses voxel representation for both input data and output heatmaps. Oasis [6] also uses 3D point cloud as input and their method is constructed based on PointNet [20]. Three different errors are used for comparison: 1) the average across the complete test set (*avg test*), 2) the average across the test set of seen subjects' hand dur-

[2] https://competitions.codalab.org/competitions/17356#learn_the_de

	our method			V2V-PoseNet [15]	Hand3D [4]	P2P-Regression [9]
GPU	GTX1060			Titan X	Titan X	Titan Xp
time		detection	regression	285.7 ms	33.3 ms	23.9 ms
	256 points	3.5 ms	2.9 ms			
	512 points	6.9 ms	5.5 ms			
	1024 points	12.5 ms	10.7 ms			

Table 4. Comparison of runtime and hardware

ing training (*seen test*), and 3) the average across the test set images of unseen subjects' hand (*unseen test*). Currently, our method achieves the lowest overall mean joint error on the test dataset of 9.82 mm. For seen subjects' hand and unseen subjects' hand, the mean joint errors are 7.15 mm and 12.04 mm respectively, which shows the generalizability of the proposed method even without regularization on the parameters. In comparison to other 3D data based methods, our method is slightly better than V2V-PoseNet, whereas V2V-PoseNet requires 10 good GPUs to run realtime and our method requires only one moderate GPU. Compared to oasis, which also uses 1024 3D points as input, our method is 1.48 mm better, where oasis requires more input information like surface normal and k-nearest neighbours.

NYU dataset. For the NYU dataset, we only compared to recent state-of-the-art methods after 2017. For testing the performance, only the frontal view was used. Following previous works [17][30][9], only 14 joints out of 36 joints provided were used for evaluation. For a fair comparison, we only compared to the methods trained solely on the NYU dataset without additional data. The compared methods include depth image based methods (DeepPrior++ [16], DenseReg [4]), 3D voxel based methods (3DCNN [8], V2V-PoseNet [15]) and point cloud based methods (SHPR-Net [3], HandPointNet [6], Point-to-Point [9]). The comparison is shown in Figure 5, where our method performs comparably good with V2V-PoseNet [15] and Point-to-Point [9], and outperforms all other methods. A closer comparison of the mean joint error value can be found in Table 3, where our method trained with single view is the second best, and our method trained with three views outperforms all recent state-of-the-art methods.

4.4. Segmentation using importance term

Besides showing the quantitative results relying on the ground-truth data, we also show some qualitative result of the segmentation using the automatically inferred importance term. As seen from Figure 6, the segmentation result is shown alongside the original point cloud. The samples are taken from the Hands2017Challenge dataset. Both samples with all visible fingers and samples with different levels of self-occlusion are shown. In all cases, the fingers are clearly segmented with each other, even the fingers are twisted together. The points has no contribution to any joint has very small importance values and they are classified as background. As Figure 6 shows, the arm and the background points are clearly segmented in gray. Notice that the segmentation result is obtained without the ground-truth data for segmentation. This leads to a future research question about whether we can perform this method on hand-object interaction cases, where the influence of the object can be automatically removed.

4.5. Runtime and model size

Compared to depth image based methods, our method requires more computation time and memory storage, this limits our training to use only 512 points (batch size=32) on our hardware setting. To store the learned models, the proposed method takes 38 MB for the regression version and 44 MB for the detection version. Compared to 420MB for a 3D CNN based method [4], our model size is much smaller. For the testing stage, the runtime of our method is 12.5 ms and 10.7 ms per frame for the detection and regression version respectively, where 1024 points are used as input. When less input points is used, the runtime can be further reduced with a small performance loss. Table 4 shows a comparison of runtime to other state-of-the-art 3D methods [15][4][9]. Although the other methods all used a more powerful GPU than ours, our method require the least processing time.

5. Conclusion

We propose to use a novel neural network architecture, ResidualPEL, for hand pose estimation using unordered point cloud as input. The proposed method is invariant to input point order and can handle different numbers of points. Compared to previous 3D voxel based methods, our method requires less memory size. And compared to PointNet based methods, our method does not require surface normal and K-nearest-neighbours information. A voting-based scheme was proposed to merge information from individual points to pose output, where the resulting importance term can be also used to segment the hand into different parts. The performance of our method is evaluated on two datasets, where our method outperforms the state-of-the-art methods on both datasets. In future work, the proposed ResidualPEL and voting scheme can be also applied to similar problem such as human pose estimation and object pose estimation.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016. 5

[2] Xinghao Chen, Guijin Wang, Hengkai Guo, and Cairong Zhang. Pose guided structured region ensemble network for cascaded hand pose estimation. *arXiv preprint arXiv:1708.03416*, 2017. 6, 7

[3] Xinghao Chen, Guijin Wang, Cairong Zhang, Tae-Kyun Kim, and Xiangyang Ji. Shpr-net: Deep semantic hand pose regression from point clouds. *IEEE Access*, 6:43425–43439, 2018. 1, 2, 6, 8

[4] Xiaoming Deng, Shuo Yang, Yinda Zhang, Ping Tan, Liang Chang, and Hongan Wang. Hand3d: Hand pose estimation using 3d neural network. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018. 1, 2, 6, 8

[5] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. *arXiv:1704.02463*, 2017. 5

[6] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. Hand pointnet: 3d hand pose estimation using point sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8417–8426, 2018. 1, 2, 6, 7, 8

[7] 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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3593–3601, 2016. 2

[8] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *IEEE conference on computer vision and pattern recognition*, pages 1991–2000, 2017. 2, 8

[9] Liuhao Ge, Zhou Ren, and Junsong Yuan. Point-to-point regression pointnet for 3d hand pose estimation. *ECCV, Springer*, 1, 2018. 1, 2, 6, 8

[10] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. Triplet-center loss for multi-view 3d object retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[11] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[12] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 863–872. IEEE, 2017. 3

[13] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[14] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 2

[15] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. *arXiv:1711.07399*, 2017. 1, 2, 6, 7, 8

[16] Markus Oberweger and Vincent Lepetit. Deepprior++: Improving fast and accurate 3d hand pose estimation. In *ICCV workshop*, volume 840, page 2, 2017. 2, 6, 8

[17] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. In *Computer Vision Winter Workshop*, pages 1–10, 2015. 2, 8

[18] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. In *IEEE International Conference on Computer Vision*, pages 3316–3324, 2015. 2

[19] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BmVC*, volume 1, page 3, 2011. 2

[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 2, 7

[21] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 2

[22] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 2

[23] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1106–1113, 2014. 2

[24] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500*, 2016. 2, 3, 4

[25] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (TOG)*, 36(6):245, 2017. 2

[26] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3633–3642. ACM, 2015. 2

[27] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *33rd ACM Conference on Human Factors in Computing Systems*, pages 3633–3642. ACM, 2015. 2

[28] Danhang Tang, Jonathan Taylor, Pushmeet Kohli, Cem Keskin, Tae-Kyun Kim, and Jamie Shotton. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *Proceedings of the IEEE international conference on computer vision*, pages 3325–3333, 2015. 2

[29] Anastasia Tkach, Andrea Tagliasacchi, Edoardo Remelli, Mark Pauly, and Andrew Fitzgibbon. Online generative model personalization for hand tracking. *ACM Transactions on Graphics (TOG)*, 36(6):243, 2017. 2

[30] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics*, 33(5):169, 2014. 2, 5, 6, 8

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. 3

[32] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 2

[33] Qi Ye and Tae-Kyun Kim. Occlusion-aware hand pose estimation using hierarchical mixture density network. *ECCV, Springer*, 2018. 2

[34] Qi Ye, Shanxin Yuan, and Tae-Kyun Kim. Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation. In *European conference on computer vision*, pages 346–361. Springer, 2016. 2

[35] Tan Yu, Jingjing Meng, and Junsong Yuan. Multi-view harmonized bilinear network for 3d object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[36] Shanxin Yuan, Guillermo Garcia-Hernando, Bjrn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, Junsong Yuan, Xinghao Chen, Guijin Wang, Fan Yang, Kai Akiyama, Yang Wu, Qingfu Wan, Meysam Madadi, Sergio Escalera, Shile Li, Dongheui Lee, Iason Oikonomidis, Antonis Argyros, and Tae-Kyun Kim. Depth-based 3d hand pose estimation: From current achievements to future goals. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 6, 7

[37] Shanxin Yuan, Qi Ye, Guillermo Garcia-Hernando, and Tae-Kyun Kim. The 2017 hands in the million challenge on 3d hand pose estimation. *arXiv:1707.02237*, 2017. 5

[38] Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhant Jain, and Tae-Kyun Kim. Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2605–2613, 2017. 5

[39] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017. 2

[40] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. Model-based deep hand pose estimation. In *Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2421–2427. AAAI Press, 2016. 2