Hand PointNet: 3D Hand Pose Estimation using Point Sets

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

Abstract

Convolutional Neural Network (CNN) has shown promising results for 3D hand pose estimation in depth images. Different from existing CNN-based hand pose estimation methods that take either 2D images or 3D volumes as the input, our proposed Hand PointNet *directly processes the 3D point cloud that models the visible surface of the hand for pose regression. Taking the normalized point cloud as the input, our proposed hand pose regression network is able to capture complex hand structures and accurately regress a low dimensional representation of the 3D hand pose. In order to further improve the accuracy of fingertips, we design a fingertip refinement network that directly takes the neighboring points of the estimated fingertip location as input to refine the fingertip location. Experiments on three challenging hand pose datasets show that our proposed method outperforms state-of-the-art methods.*

1. Introduction

Recent years have witnessed a steady growth of the research in real-time 3D hand pose estimation with depth cameras [12, 45, 38, 18, 36, 31, 8, 3, 48], since this technology can improve user experience and play an important role in various human-computer interaction applications, especially in virtual reality and augmented reality applications. However, due to the high dimensionality of 3D hand pose, large variations in hand orientations, high self-similarity of fingers and severe self-occlusion, 3D hand pose estimation still suffers from the issues of accuracy and robustness.

With the success of deep neural networks in various computer vision tasks and the emergence of large hand pose datasets [38, 34, 33, 49, 48], many of the recent 3D hand pose estimation methods are based on CNNs [38, 21, 7, 8, 9, 3, 19, 51]. Considering 2D CNNs that take 2D images as input cannot fully utilize 3D spatial information in the depth image, Ge et al. [8] encodes the hand depth images as 3D volumes and applies a 3D CNN for inferring 3D hand pose. However, the time and space complexities of the 3D CNN grow cubically with the resolution of the input 3D volume [27]. Consequently, the 3D volume adopted in [8] is limited to a low resolution (e.g., 32^3), which may lose useful details of the hand. Furthermore, due to the sparsity of 3D point cloud, most of the voxels in the 3D volume are usually not occupied by any points, which not only wastes computations of 3D convolutions, but also distracts the neural networks from learning effective kernels to capture meaningful features of hand shapes. The approach in [8] transforms the sparse point cloud to a dense volumetric representation to enable effective 3D convolution. But this transformation changes the nature of the data and makes the data unnecessarily voluminous.

To tackle these problems, motivated by the recent works of PointNet [23, 25] that perform 3D object classification and segmentation on point sets directly, we aim at learning 3D hand articulations directly from the 3D point cloud instead of rasterizing the 3D points into 3D voxels. It is worth noting that the depth image in essence is represented by a set of unordered 3D points on the visible surface of the hand, which is essentially 2.5D data and is not directly suitable to be processed by 2D or 3D convolutions. Compared with previous multi-view CNNs-based method [7] and 3D CNN-based method [8], our approach does not need to project the point cloud into multiple 2D images or transform the sparse point cloud into dense 3D volumes, thus can better utilize the original point cloud in an effective and efficient way.

In this work, we propose a point cloud based hand joints regression method for 3D hand pose estimation in single depth images, as illustrated in Figure 1. Specifically, the segmented hand depth image is first converted to a set of 3D points; the 3D point cloud of the hand is downsampled and normalized in an oriented bounding box to make our method robust to various hand orientations. The hierarchical PointNet [25] takes the 3D coordinates of normalized points attached with the estimated surface normals as the input, and outputs a low dimensional representation of the 3D hand joint locations which are then recovered in the

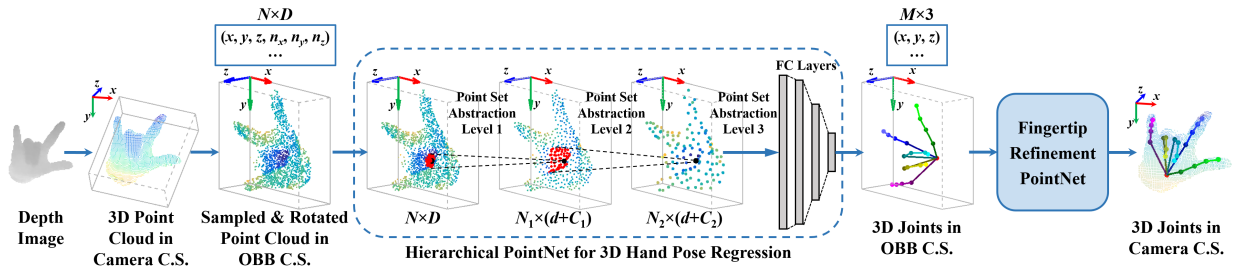

Figure 1: Overview of our proposed Hand PointNet-based method for 3D hand pose estimation in single depth images. We normalize the 3D point cloud in an oriented bounding box (OBB) to make the network input robust to global hand rotation. The 3D coordinates of sampled and normalized points attached with estimated surface normals are fed into a hierarchical PointNet [25], which is trained in an end-to-end manner, to extract hand features and regress 3D joint locations. The fingertip refinement PointNet can further improve the estimation accuracy of fingertip locations.

camera coordinate system. The fingertip locations are further refined by a basic PointNet which takes the neighboring points of the estimated fingertip location as input. To our knowledge, this is the first work that regresses 3D hand joint locations directly from the 3D point cloud using a deep neural network, which is able to capture 3D structures of the hand and estimate 3D hand pose accurately in real-time. Our main contributions are summarized as follows:

- We propose to estimate 3D hand joint locations directly from 3D point cloud based on the network architecture of PointNet [23, 25]. Compared with 2D CNN-based and multi-view CNNs-based methods [38, 7, 9, 3, 19], our method can better exploit the 3D spatial information in the depth image; compared with 3D CNN-based method [8], our method can effectively leverage more information in the depth image to capture more details of the hand with fewer number of network parameters.
- In order to make our method robust to variations in hand global orientations, we propose to normalize the sampled 3D points in an oriented bounding box without applying any additional network to transform the hand point cloud. The normalized point clouds with more consistent global orientations make the PointNet easier to learn 3D hand articulations.
- We propose to refine the fingertip locations with a basic PointNet that takes the neighboring points of the estimated fingertip location as input to regress the refined fingertip location. The refinement network can further exploit finer details in the original point cloud and regress more accurate fingertip locations.

We conduct comprehensive experiments on three challenging hand pose datasets [33, 34, 38] to evaluate our methods. Experimental results show that our proposed Hand PointNet-based method for 3D hand pose estimation is superior to state-of-the-art methods on all the three datasets, with runtime speed of over 48fps.

2. Related Work

Hand Pose Estimation: Hand pose estimation methods can be divided into discriminative approaches [12, 45, 16, 38], generative approaches [22, 1, 39, 13, 26] and hybrid approaches [28, 36, 35]. In this section, we focus on deep neural networks-based discriminative approaches.

Some sophisticated neural networks have been explored for 3D hand pose estimation, such as the feedback loop model [21], spatial attention network [47], region ensemble network [9], deep generative models [41], *etc*. Hand joint constraints are integrated into deep learning approaches to avoid implausible hand poses. Oberweger et al. [20, 19] exploit a hand prior to constrain the hand pose. Zhou et al. [50] train a CNN to regress hand model parameters and infer hand pose via forward kinematics. Some other methods focus on different input representations. Ge et al. [7, 8] apply multi-view CNNs and 3D CNN for 3D hand pose estimation which take projected images on multiple views and 3D volumes as input, respectively, in order to better utilize the depth information. Choi et al. [3] adopt geometric features as additional input modality to estimate hand poses through multi-task learning. However, none of these methods directly take the point cloud as the neural network input.

3D Deep Learning: Multi-view CNNs-based approaches [32, 24, 7, 2] project 3D points into 2D images and use 2D CNNs to process them. 3D CNN-based methods [44, 17, 24, 30, 8] rasterize 3D points into 3D voxels for 3D convolution. 3D CNNs based on octrees [27, 43] are proposed for efficient computation on high resolution volumes.

PointNet [23, 25] is a recently proposed method that directly takes point cloud as network input. Similar to PointNet, deep Kd-networks [15] is a recently proposed network architecture that directly consumes point cloud by adopting a Kd-tree structure. These methods have shown promising performance on 3D classification and segmentation tasks, but have not been applied to articulated pose regression.

3. Methodology

Our proposed 3D hand pose estimation method takes a depth image containing a hand as the input and outputs a set of 3D hand joint locations $\mathbf{\Phi} = \{\boldsymbol{\phi}_m\}_{m=1}^{M} \in \mathbf{\Lambda}$ in the camera coordinate system (C.S.), where M is the number of hand joints, $\mathbf{\Lambda}$ is the $3 \times M$ dimensional hand joint space.

The hand depth image is converted to a set of 3D points. The 3D point set is downsampled to N points $\boldsymbol{p}_i \in \mathbb{R}^3$ $(i = 1, \cdots, N)$, and normalized in an oriented bounding box (OBB). A hierarchical PointNet [25] takes N points as the input to extract hierarchical hand features and regress the 3D hand pose. The dimension of each input point is $D = d + C_0$, which is composed of d-dim coordinate and C_0-dim input feature. In this work, $d = 3$, and $C_0 = 3$ since we adopt the estimated 3D surface normal as the input feature. To further improve the estimation accuracy of fingertip locations, a fingertip refinement network is designed. In the following sections, we first briefly review the mechanism of PointNet, then present our proposed 3D hand pose estimation method.

3.1. PointNet Revisited

Basic PointNet: PointNet [23] is a type of neural network that directly takes a set of points as the input and is able to extract discriminative features of the point cloud. As shown in Figure 2, each input point $\boldsymbol{x}_i \in \mathbb{R}^D$ $(i = 1, \cdots, N)$ is mapped into a C-dim feature vector through multi-layer perceptron (MLP) networks, of which the weights across different points are shared. A vector max operator is applied to aggregate N point feature vectors into a global feature vector that is invariant to permutations of input points. Finally, the C-dim global feature vector is mapped into an F-dim output vector using MLP networks. It has been proved in [23] that PointNet has the ability to approximate arbitrary continuous set functions, given enough neurons in the network.

Hierarchical PointNet: The main limitation of the basic PointNet is that it cannot capture local structures of the point cloud in a hierarchical way. To address this problem, Qi et al. [25] proposed a hierarchical PointNet which has better generalization ability due to its hierarchical feature extraction architecture. In this paper, we exploit the hierarchical PointNet for 3D hand pose estimation. As shown in Figure 1, the hierarchical PointNet consists of L point set abstraction levels. At the l-th level $(l = 1, \cdots, L-1)$, N_l points are selected as centroids of local regions; the k-nearest neighbors of the centroid point are grouped as a local region; a basic PointNet with shared weights across different local regions is applied to extract a C_l-dim feature of each local region; N_l centroid points with d-dim coordinates and C_l-dim features are fed into the next level. At the last level, a global point cloud feature is abstracted from the whole input points of this level by using a basic PointNet.

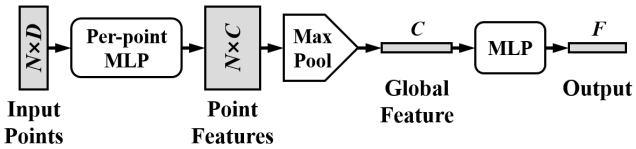

Figure 2: Basic architecture of PointNet. The network directly takes N points as the input. Each D-dim input point is mapped into a C-dim feature through MLP. Per-point features are aggregated into a global feature by max-pooling. The global feature is mapped into an F-dim output vector.

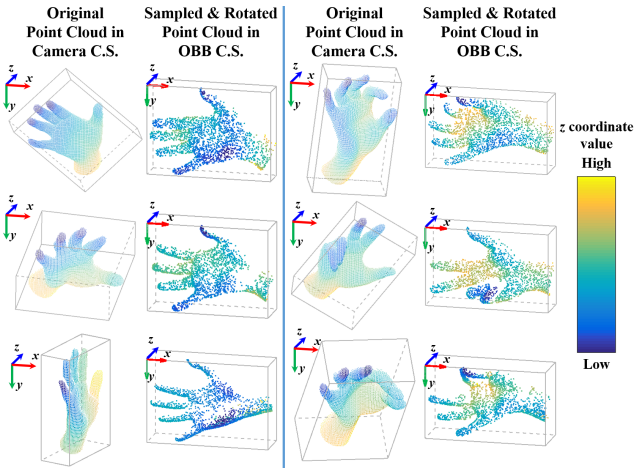

Figure 3: Examples of OBB-based point cloud normalization. The 1st and 3rd columns present the original point clouds in the camera C.S. with various hand orientations. The 2nd and 4th columns present the sampled and rotated point clouds in the OBB C.S., of which the hand orientations are more consistent. We color-code all point clouds to show the relative z coordinate value.

3.2. OBB-based Point Cloud Normalization

One challenge of 3D hand pose estimation is the large variation in global orientation of the hand. The objective for hand point cloud normalization is to transform the original hand point cloud into a canonical coordinate system in which the global orientations of the transformed hand point clouds are as consistent as possible. This normalization step ensures that our method is robust to variations in hand global orientations.

If the output is invariant to the rotation of input point cloud, such as the outputs in semantic labeling tasks [23], we can add a spatial transformation network [11, 23] into the PointNet to predict the transformation matrix, as proposed in [23]. However, in our problem, the output 3D hand pose depends on the orientation of the input hand point cloud, which makes the network difficult to be trained in an

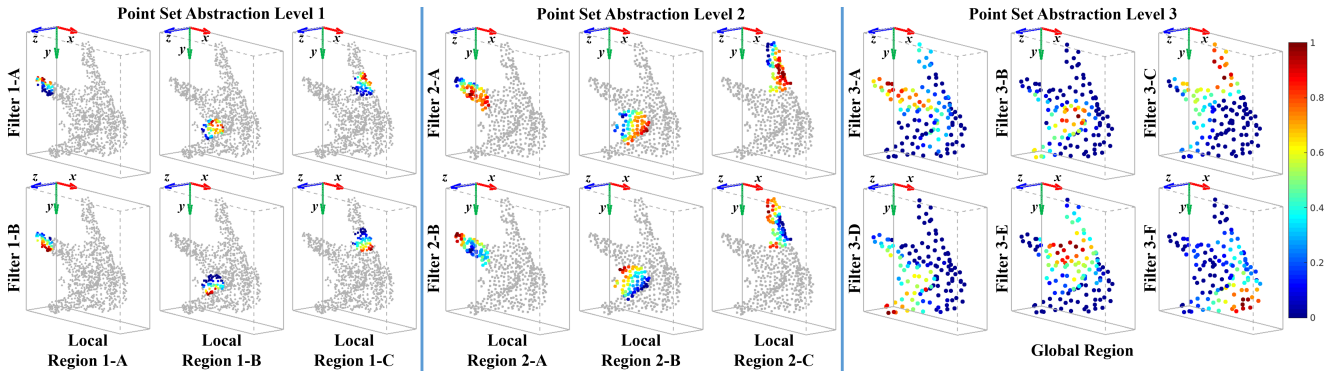

Figure 4: Visualization of the point sensitivity in different regions to different filters at three set abstraction levels. At each of the first two levels, each column corresponds to the same local region, and each row corresponds to the same filter. For illustration purpose, we only show the sensitivity of points in three local regions to two filters at each of the first two levels; At the third level, we show the sensitivity of points to six filters. Points with high sensitivity are shown in red color, and points with low sensitivity are shown in blue color. Points that do not belong to the local region are shown in gray color.

end-to-end manner. Some solutions could be designing two parallel networks to estimate the canonical hand pose and the hand orientation at the same time [51] or adding a spatial de-transformation network before the output layer to remap the estimated pose to the original C.S. [5]. But these solutions will increase the complexity of the network and make the training more difficult.

In this work, we propose a simple yet effective method to normalize the 3D hand point cloud in OBB, instead of applying any additional networks to estimate the hand global orientation or transform the hand point cloud. OBB is a tightly fitting bounding box of the input point cloud [40]. The orientation of OBB is determined by performing principal component analysis (PCA) on the 3D coordinates of input points. The x, y, z axes of the OBB C.S. are aligned with the eigenvectors of input points' covariance matrix, which correspond to eigenvalues from largest to smallest, respectively. The original points in camera C.S. are first transformed into OBB C.S., then these points are shifted to have zero mean and scaled to a unit size:

$$\begin{aligned} \boldsymbol{p}^{obb} &= (\boldsymbol{R}_{obb}^{cam})^T \cdot \boldsymbol{p}^{cam}, \\ \boldsymbol{p}^{nor} &= \left(\boldsymbol{p}^{obb} - \bar{\boldsymbol{p}}^{obb}\right) / L_{obb}, \end{aligned} \tag{1}$$

where $\boldsymbol{R}_{obb}^{cam}$ is the rotation matrix of the OBB in camera C.S.; $\boldsymbol{p}^{cam}$ and $\boldsymbol{p}^{obb}$ are 3D coordinates of point $\boldsymbol{p}$ in camera C.S. and OBB C.S., respectively; $\bar{\boldsymbol{p}}^{obb}$ is the centroid of point cloud $\{\boldsymbol{p}_i^{obb}\}_{i=1}^N$; L_{obb} is the maximum edge length of OBB; $\boldsymbol{p}^{nor}$ is the normalized 3D coordinate of point $\boldsymbol{p}$ in the normalized OBB C.S.

During training, the ground truth 3D joint locations in camera C.S. also apply the transformation in Equation 1 to obtain the 3D joint locations in the normalized OBB C.S. During testing, the estimated 3D joint locations in the normalized OBB C.S. $\hat{\boldsymbol{\phi}}_m^{nor}$ are transformed back to those in camera C.S. $\hat{\boldsymbol{\phi}}_m^{cam}$ $(m = 1, \cdots, M)$:

$$\hat{\boldsymbol{\phi}}_m^{cam} = \boldsymbol{R}_{obb}^{cam} \cdot \left(L_{obb} \cdot \hat{\boldsymbol{\phi}}_m^{nor} + \bar{\boldsymbol{p}}^{obb}\right). \tag{2}$$

Figure 3 presents some examples of the original hand point clouds and the corresponding normalized hand point clouds. As can be seen, although the orientations of original hand point clouds in camera C.S. have large variations, the orientations of normalized hand point cloud in OBB C.S. are more consistent. Experiments are conducted in Section 4.1 to show that our proposed OBB-based point cloud normalization method can improve the performance of both the basic PointNet and the hierarchical PointNet.

3.3. Hand Pose Regression Network

We design a 3D hand pose regression network which can be trained in an end-to-end manner. The input of the hand pose regression network is a set of normalized points $\boldsymbol{X}^{nor} = \{\boldsymbol{x}_i^{nor}\}_{i=1}^N = \{(\boldsymbol{p}_i^{nor}, \boldsymbol{n}_i^{nor})\}_{i=1}^N$, where $\boldsymbol{p}_i^{nor}$ is the 3D coordinate of the normalized point and $\boldsymbol{n}_i^{nor}$ is the corresponding 3D surface normal. These N points are then fed into a hierarchical PointNet [25], as shown in Figure 1, which has three point set abstraction levels. The first two levels group input points into $N_1 = 512$ and $N_2 = 128$ local regions, respectively. Each local region contains $k = 64$ points. These two levels extract $C_1 = 128$ and $C_2 = 256$ dimensional features for each local region, respectively. The last level extracts a 1024-dim global feature vector which is mapped to an F-dim output vector by three fully-connected layers. We present the detailed hierarchical PointNet architecture in the supplementary material.

Since the degree of freedom of human hand is usually lower than the dimension of 3D hand joint locations

$(3 \times M)$, the PointNet is designed to output an F-dim $(F < 3 \times M)$ representation of hand pose to enforce hand pose constraint and alleviate infeasible hand pose estimations, which is similar to [20]. In the training phase, given T training samples with the normalized point cloud and the corresponding ground truth 3D joint locations $\{(\boldsymbol{X}_t^{nor}, \boldsymbol{\Phi}_t^{nor})\}_{t=1}^{T}$, we minimize the following objective function:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{t=1}^{T} \|\boldsymbol{\alpha}_t - \mathcal{F}(\boldsymbol{X}_t^{nor}, \boldsymbol{w})\|^2 + \lambda \|\boldsymbol{w}\|^2 \quad (3)$$

where $\boldsymbol{w}$ denotes network parameters; $\mathcal{F}$ represents the hand pose regression PointNet; λ is the regularization strength; $\boldsymbol{\alpha}_t$ is an F-dim projection of $\boldsymbol{\Phi}_t^{nor}$. By performing PCA on the ground truth 3D joint locations in the training dataset, we can obtain $\boldsymbol{\alpha}_t = \boldsymbol{E}^T \cdot (\boldsymbol{\Phi}_t^{nor} - \boldsymbol{u})$, where $\boldsymbol{E}$ denotes the principal components, and $\boldsymbol{u}$ is the empirical mean. During testing, the estimated 3D joint locations are reconstructed from the network outputs:

$$\hat{\boldsymbol{\Phi}}^{nor} = \boldsymbol{E} \cdot \mathcal{F}(\boldsymbol{X}^{nor}, \boldsymbol{w}^*) + \boldsymbol{u}. \quad (4)$$

In Figure 4, we visualize the sensitivity of points in different regions to different filters at three set abstraction levels in a well trained three-level hierarchical PointNet. The corresponding input is the example shown in Figure 1. We normalize per-point features in the same region extracted by the same filter between 0 and 1. Large normalized feature value means that the corresponding point is sensitive to the filter. As can be seen in Figure 4, from low level to high level, the size of receptive field in Euclidean space becomes larger and larger. At the first two levels, filters extract features from local regions, and different filters exaggerate different local structures in the local region. At the highest level, filters extract features from the whole input points of this level, and different filters exaggerate different parts of the hand, such as different fingers, hand palm, *etc*. With such a hierarchical architecture, the network can capture structures of the hand from local to global.

3.4. Fingertip Refinement Network

To further improve the estimation accuracy of fingertip locations, we design a fingertip refinement network which takes K nearest neighboring points of the estimated fingertip location as input and outputs the refined 3D location of the fingertip, as shown in Figure 5. We refine fingertip locations for two reasons: first, the estimation error of fingertip locations is usually relatively large compared to other joints, as shown in experimental results in Section 4.1; second, the fingertip location of straightened finger is usually easy to be refined, since the K nearest neighboring points of the fingertip will not change a lot even if the estimated location deviates from the ground truth location to some extent when K is relatively large.

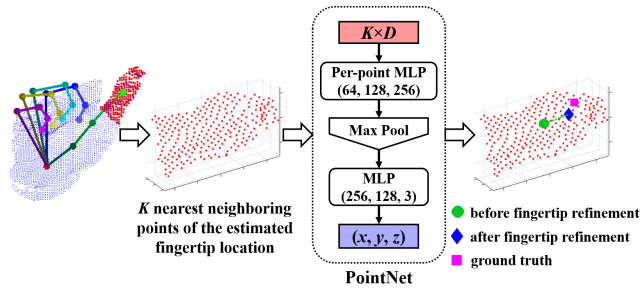

Figure 5: Illustration of fingertip refinement. K nearest neighboring points of the estimated fingertip location are found in the original hand point cloud with upper limit of point number. These points are normalized and fed into a basic PointNet that outputs the refined fingertip 3D location. Numbers in parentheses of MLP networks are layer sizes.

Since we only refine fingertips for straightened fingers, we first check each finger is bent or straightened by calculating joint angles using the joint locations. For the straightened finger, we find the K nearest neighboring points of the fingertip location in the original point cloud with upper limit of point number to ensure real-time performance. The K nearest neighboring points are then normalized in OBB, which is similar to the method in Section 3.2. A basic PointNet takes these normalized points as input and outputs the refined fingertip 3D location. During the training stage, we use the ground truth joint locations to calculate joint angles; for the fingertip location used in the nearest neighbor search, we add a 3D random offset within a radius of $r = 15mm$ to the ground truth fingertip location in order to make the fingertip refinement network more robust to inaccurate fingertip estimations. During the testing stage, we use joint locations estimated by the hand pose regression network for calculating joint angles and searching nearest neighboring points.

3.5. Implementation Details

The 3D surface normal is approximated by performing PCA on the nearest neighboring points of the query point in the sampled point cloud to fit a local plane [10]. The number of nearest neighboring points is set as 30. For the hierarchical PointNet, followed by [25], we use farthest point sampling method to sample centroids of local regions and ball query to group points. The radius for ball query is set as 0.1 at the first level and 0.2 at the second level. The output dimension of the hand pose regression network F is set as $2 \times M$, which is $2/3$ of the dimension of the hand joint locations.

For training PointNets, we use Adam [14] optimizer with initial learning rate 0.001, batch size 32 and regularization strength 0.0005. The learning rate is divided by 10 after 50

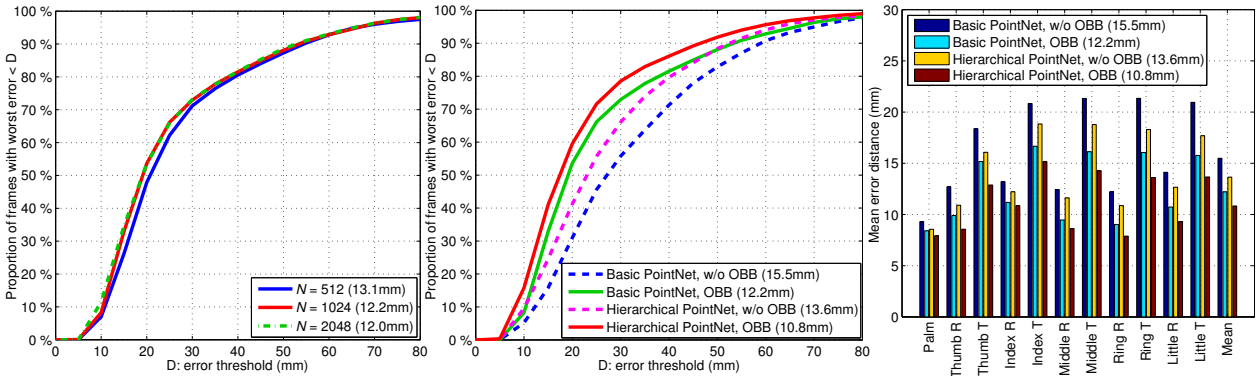

Figure 6: Self-comparison of different methods on NYU dataset [38]. **Left**: the impact of different numbers of sampled points on the proportion of good frames. **Middle & Right**: the impact of different PointNet architectures and normalization methods on the proportion of good frames as well as on the per-joint mean error distance (R: root, T: tip). The overall mean error distances are shown in parentheses.

epochs. The training process is stopped after 60 epochs.

For fingertip refinement, we set the number of nearest neighboring points K as 256. Considering the real-time performance, we downsample the original point cloud to $N' = 6000$ points with random sampling when the number of points in the original point cloud exceeds the upper limit N', and we apply Kd-tree algorithm [6] in the efficient nearest neighbor search.

4. Experiments

We evaluate our proposed method on three public hand pose datasets: NYU [38], MSRA [33] and ICVL [34].

The NYU dataset [38] contains more than 72K training frames and 8K testing frames. Each frame contains 36 annotated joints. Following previous works [38, 21, 8], we estimate a subset of $M = 14$ joints. We segment the hand from the depth image using random decision forest (RDF) similar to [34]. Since the segmented hands may contain arms with various lengths, we augment the training data with random arm lengths.

The MSRA dataset [33] contains more than 76K frames from 9 subjects. Each subject contains 17 gestures. In each frame, the hand has been segmented from the depth image and the ground truth contains $M = 21$ joints. The neural networks are trained on 8 subjects and tested on the remaining subject. We repeat this experiment 9 times for all subjects and report the average metrics. We do not perform any data augmentation on this dataset.

The ICVL dataset [34] contains 22K training frames and 1.6K testing frames. The ground truth of each frame contains $M = 16$ joints. We apply RDF for hand segmentation and augment the training data with random arm lengths as well as random stretch factors.

We evaluate the hand pose estimation performance with two metrics: the first metric is the per-joint mean error distance over all test frames; the second metric is the proportion of good frames in which the worst joint error is below a threshold, which is proposed in [37] and is more strict.

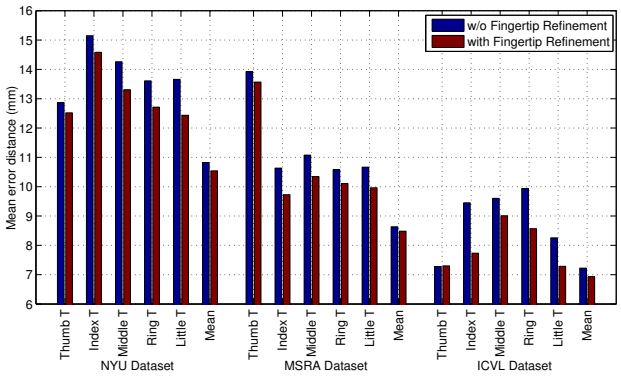

Figure 7: The impact of fingertip refinement on fingertips' mean error distances and the overall mean error distance on NYU [38], MSRA [33] and ICVL [34] datasets (T: tip).

All experiments are conducted on a workstation with two Intel Core i7 5930K, 64GB of RAM and an Nvidia GTX1080 GPU. The deep neural networks are implemented within the PyTorch framework.

4.1. Self-comparisons

We first evaluate the influence of the number of sampled points N. As shown in Figure 6 (left), we experiment with three different numbers of sampled points by using a basic PointNet for hand pose regression on NYU dataset. When $N = 512$, the estimation accuracy is slightly lower than the other two results with larger number of sampled

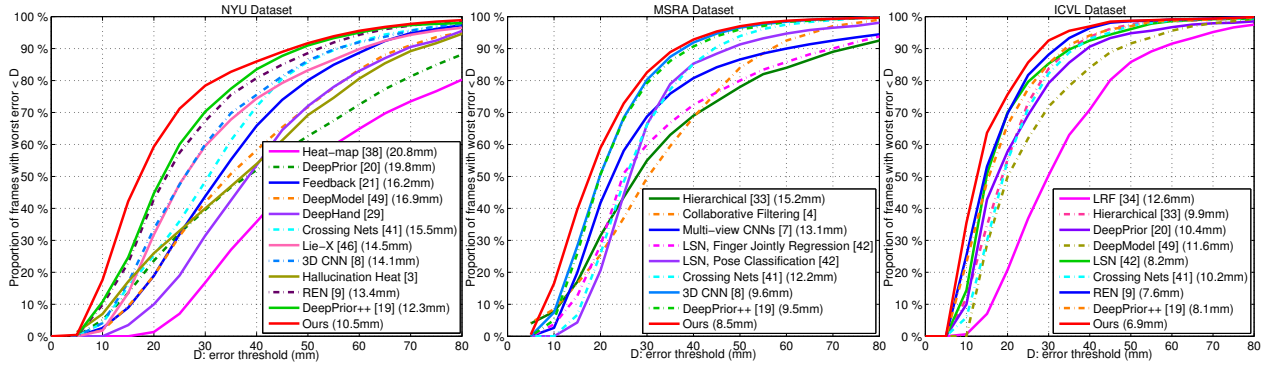

Figure 8: Comparison with state-of-the-art methods on NYU [38] (left), MSRA [33] (middle) and ICVL [34] (right) datasets. The proportions of good frames and the overall mean error distances (in parentheses) are presented in this figure.

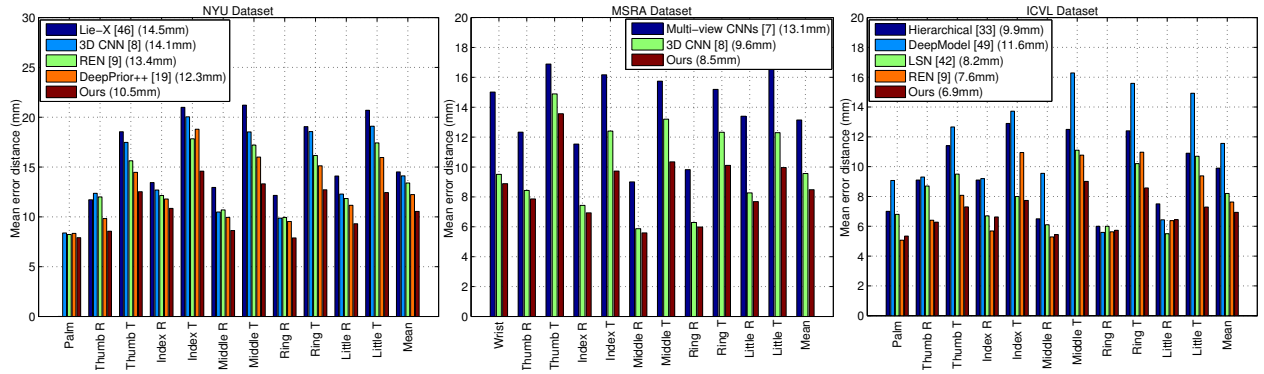

Figure 9: Comparison with state-of-the-art methods on NYU [38] (left), MSRA [33] (middle) and ICVL [34] (right) datasets. The per-joint mean error distances and the overall mean error distances are presented in this figure (R: root, T: tip).

points. But when $N = 1024$ and $N = 2048$, the estimation accuracy is almost the same. Balancing between estimation accuracy and real-time performance, we choose $N = 1024$ and apply this number of sampled points in the following experiments. This experiment also shows that our method is robust to a small amount of sampled points, since the performance does not drop a lot when $N = 512$.

We evaluate the impact of different PointNet architectures and normalization methods on NYU dataset without fingertip refinement. Note that, for the normalization method without using OBB, we shift the point cloud center to zero and scale the points into a unit sphere without rotating the point cloud. As presented in Figure 6 (middle and right), with the same PointNet architecture, the estimation accuracy of OBB-based point cloud normalization method is superior to that of the normalization method without using OBB by a large margin, which indicates that our proposed OBB-based point cloud normalization method is quite effective since it can normalize point cloud with more consistent orientations and make the network easier to learn the hand articulations. In addition, when adopting the same normalization method, the hierarchical PointNet performs better than the basic PointNet on the estimation accuracy, which shows that the hierarchical network architecture [25] is also effective in our hand joints regression task.

In addition, we study the influence of fingertip refinement. As can be seen in Figure 7, after fingertip refinement, the average estimation errors of most fingertips as well as the overall mean error decrease on all the three datasets, which indicates that our proposed fingertip refinement method can further improve the estimation accuracy of fingertip locations.

4.2. Comparisons with State-of-the-arts

We compare our point cloud based hand joints regression method with 16 state-of-the-art methods: latent random forest (LRF) [34], hierarchical regression with random forest (Hierarchical) [33], collaborative filtering [4], 2D CNN for heat-map regression (Heat-map) [38], 2D CNN with prior and refinement (DeepPrior) [20], 2D CNN with feed-

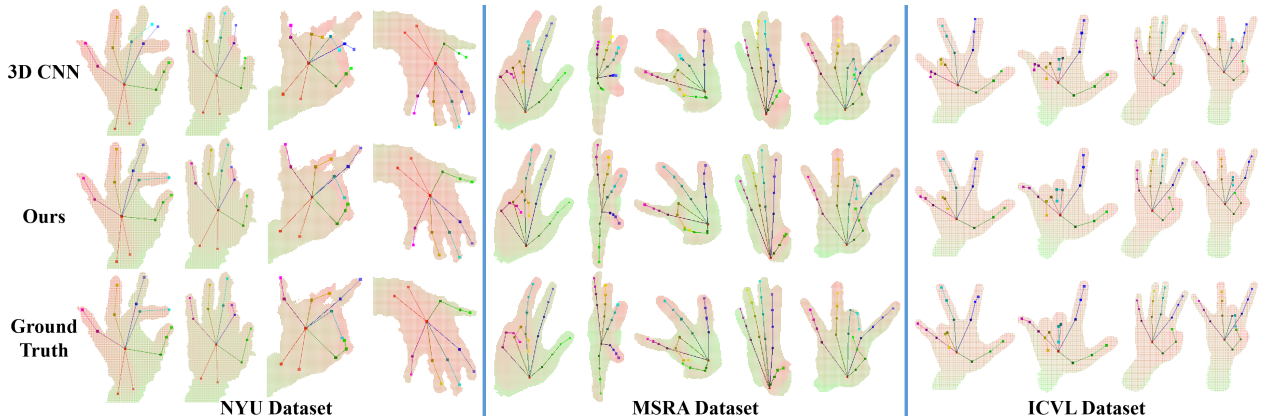

Figure 10: Qualitative results for NYU [38] (left), MSRA [33] (middle) and ICVL [34] (right) datasets. We compare our method (in the second row) with the 3D CNN-based method in [8] (in the first row). The ground truth hand joint locations are presented in the last row. We show hand joint locations on depth images. Different hand joints and bones are visualized with different colors. This figure is best viewed in color.

back loop (Feedback) [21], 2D CNN for hand model regression (DeepModel) [50], matrix completion with deep feature (DeepHand) [29], local surface normal based random forest (LSN) [42], multi-view CNNs [7], 2D CNN using Lie group theory (Lie-X) [46], Crossing Nets [41], 3D CNN [8], region ensemble network (REN) [9], improved DeepPrior (DeepPrior++) [19] and 2D CNN with hallucinating heat distribution (Hallucination) [3]. The proportion of good frames over different error thresholds and the per-joint mean error distances of different methods on NYU [38], MSRA [33] and ICVL [34] datasets are presented in Figure 8 and Figure 9, respectively.

As can be seen in Figure 8, our method outperforms these state-of-the-art methods over most of the error thresholds. On NYU dataset [38], when the error threshold is between 15mm and 20mm, the proportions of good frames of our method is about 15% better than REN [9] and DeepPrior++ [19] methods. On MSRA dataset [33], when the error threshold is between 15mm and 20mm, the proportions of good frames of our method is about 10% better than 3D CNN [8] and DeepPrior++ [19] methods. On ICVL dataset [34], when the error threshold is 15mm, the proportions of good frames of our method is about 15% better than REN [9] method and 10% better than DeepPrior++ [19] method. For the mean error distances shown in Figure 9, our method also outperforms state-of-the-art methods on most of the hand joints and achieves the smallest overall mean error distances on all the three datasets.

Some qualitative results for NYU [38], MSRA [33] and ICVL [34] datasets are shown in Figure 10. As can be seen, compared with the 3D CNN-based method in [8], our method can capture complex hand structures and estimate more accurate 3D hand joint locations.

4.3. Runtime and Model Size

The runtime of our method is 20.5ms in average, including 8.2ms for point sampling and surface normal calculation, 9.2ms for the hand pose regression network forward propagation, 2.8ms for fingertip neighboring points search and 0.3ms for fingertip refinement network forward propagation. Thus, our method runs in real-time at over 48fps.

In addition, our network model size is 10.3MB, including 9.2MB for the hand pose regression network and 1.1MB for the fingertip refinement network, while the model size of the 3D CNN in [8] is about 420MB.

5. Conclusion

In this paper, we present a novel approach that directly takes point cloud as network input to regress 3D hand joint locations. To handle variations of hand global orientations, we normalize the hand point clouds in OBB with more consistent global orientations. The normalized point cloud is then fed into a hierarchical PointNet for hand pose regression. Fingertip locations are further refined by a basic PointNet. Experimental results on three public hand pose datasets show that our method achieves superior performance for 3D hand pose estimation in real-time.

Acknowledgment: This research is supported by the Interdisciplinary Graduate School, NTU. This work is also supported by the BeingTogether Centre, a collaboration between NTU Singapore and UNC at Chapel Hill. The BeingTogether Centre is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

References

[1] L. Ballan, A. Taneja, J. Gall, L. V. Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points. In *ECCV*, 2012.

[2] Z. Cao, Q. Huang, and K. Ramani. 3d object classification via spherical projections. In *3DV*, 2017.

[3] C. Choi, S. Kim, and K. Ramani. Learning hand articulations by hallucinating heat distribution. In *ICCV*, 2017.

[4] C. Choi, A. Sinha, J. Hee Choi, S. Jang, and K. Ramani. A collaborative filtering approach to real-time hand pose estimation. In *ICCV*, 2015.

[5] H. Fang, S. Xie, Y.-W. Tai, and C. Lu. RMPE: Regional multi-person pose estimation. In *ICCV*, 2017.

[6] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[7] L. Ge, H. Liang, J. Yuan, and D. Thalmann. Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs. In *CVPR*, 2016.

[8] 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.

[9] 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.

[10] H. Hoppe, T. DeRose, T. Duchamp, J. Mcdonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer graphics*, 26(2):71–78, 1992.

[11] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015.

[12] C. Keskin, F. Kra, Y. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *ECCV*, 2012.

[13] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *CVPR*, 2015.

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[15] R. Klokov and V. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *ICCV*, 2017.

[16] H. Liang, J. Yuan, and D. Thalmann. Resolving ambiguous hand pose predictions by exploiting part correlations. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(7):1125–1139, 2015.

[17] D. Maturana and S. Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In *IROS*, 2015.

[18] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural network. In *CVPR*, 2016.

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

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

[21] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a feedback loop for hand pose estimation. In *ICCV*, 2015.

[22] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *BMVC*, 2011.

[23] 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.

[24] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, 2016.

[25] 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.

[26] E. Remelli, A. Tkach, A. Tagliasacchi, and M. Pauly. Low-dimensionality calibration through local anisotropic scaling for robust hand model personalization. In *ICCV*, 2017.

[27] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017.

[28] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi. Accurate, robust, and flexible real-time hand tracking. In *CHI*, 2015.

[29] A. Sinha, C. Choi, and K. Ramani. Deephand: Robust hand pose estimation by completing a matrix with deep features. In *CVPR*, 2016.

[30] S. Song and J. Xiao. Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In *CVPR*, 2016.

[31] S. Sridhar, F. Mueller, M. Zollhoefer, D. Casas, A. Oulasvirta, and C. Theobalt. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *ECCV*, 2016.

[32] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015.

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

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

[35] D. Tang, J. Taylor, P. Kohli, C. Keskin, T.-K. Kim, and J. Shotton. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *ICCV*, 2015.

[36] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, S. Izadi, R. Banks, A. Fitzgibbon, and J. Shotton. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics*, 35(4):143, 2016.

[37] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *CVPR*, 2012.

[38] 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.

[39] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision*, 118(2):172–193, 2016.

[40] J. M. Van Verth and L. M. Bishop. *Essential mathematics for games and interactive applications*. CRC Press, 2015.

[41] C. Wan, T. Probst, L. Van Gool, and A. Yao. Crossing nets: Dual generative models with a shared latent space for hand pose estimation. In *CVPR*, 2017.

[42] C. Wan, A. Yao, and L. Van Gool. Direction matters: hand pose estimation from local surface normals. In *ECCV*, 2016.

[43] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):72, 2017.

[44] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[45] C. Xu and L. Cheng. Efficient hand pose estimation from a single depth image. In *ICCV*, 2013.

[46] C. Xu, L. N. Govindarajan, Y. Zhang, and L. Cheng. Lie-X: Depth image based articulated object pose estimation, tracking, and action recognition on lie groups. *International Journal of Computer Vision*, 123(3):454–478, 2017.

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

[48] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Y. Chang, K. M. Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, J. Yuan, X. Chen, G. Wang, F. Yang, K. Akiyama, Y. Wu, Q. Wan, M. Madadi, S. Escalera, S. Li, D. Lee, I. Oikonomidis, A. Argyros, and T.-K. Kim. Depth-based 3D hand pose estimation: From current achievements to future goals. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[49] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim. Bighand2.2m benchmark: Hand pose dataset and state of the art analysis. In *CVPR*, 2017.

[50] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei. Model-based deep hand pose estimation. In *IJCAI*, 2016.

[51] C. Zimmermann and T. Brox. Learning to estimate 3D hand pose from single RGB images. In *ICCV*, 2017.