Point-to-Point Regression PointNet for 3D Hand Pose Estimation

Liuhao Ge[1], Zhou Ren[2], and Junsong Yuan[3]

[1] Institute for Media Innovation, Interdisciplinary Graduate School,
Nanyang Technological University, Singapore
`ge0001ao@e.ntu.edu.sg`
[2] Snap Inc., 64 Market Street, Venice, CA, USA
`zhou.ren@snapchat.com`
[3] Department of Computer Science and Engineering,
State University of New York at Buffalo, NY, USA
`jsyuan@buffalo.edu`

Abstract. Convolutional Neural Networks (CNNs)-based methods for 3D hand pose estimation with depth cameras usually take 2D depth images as input and directly regress holistic 3D hand pose. Different from these methods, our proposed *Point-to-Point Regression PointNet* directly takes the 3D point cloud as input and outputs point-wise estimations, *i.e.*, heat-maps and unit vector fields on the point cloud, representing the closeness and direction from every point in the point cloud to the hand joint. The point-wise estimations are used to infer 3D joint locations with weighted fusion. To better capture 3D spatial information in the point cloud, we apply a stacked network architecture for PointNet with intermediate supervision, which is trained end-to-end. Experiments show that our method can achieve outstanding results when compared with state-of-the-art methods on three challenging hand pose datasets.

Keywords: 3D Hand Pose Estimation

1 Introduction

A key technology for human-computer interaction in virtual reality and augmented reality applications is accurate and real-time 3D hand pose estimation, which allows direct hand interaction with virtual objects. Despite the recent progress of 3D hand pose estimation with depth cameras [23, 13, 51, 38, 43, 36, 35, 11, 22, 45, 54, 17], it remains challenging to achieve accurate and robust results due to the high dimensionality and large variations of 3D hand pose, high similarity among fingers, severe self-occlusion, and noisy depth images.

Most of the recently proposed 3D hand pose estimation methods [11, 22, 45, 10, 53, 12, 19, 4, 5] are based on convolutional neural networks (CNNs) and have achieved drastic performance improvement on large hand pose datasets [43, 36, 35, 55]. Many methods directly regress 3D coordinates of hand joints or hand pose parameters using CNNs [7, 11, 9, 22, 45, 12, 19, 4, 5, 21, 56]. However, the direct mapping from input representation to 3D hand pose is highly non-linear and difficult to learn, which makes these direct regression methods difficult to achieve high accuracy [42]. An alternative way is to generate a set of heat-maps representing the probability distributions of joint locations on 2D image plane [43, 10, 8], which has been successfully applied in 2D human pose estimation [49, 18]. However, it is non-trivial to lift 2D heat-maps to 3D joint locations [24, 30, 41]. One straightforward solution is to generate volumetric heat-maps using 3D CNNs, but it is computationally inefficient. Wan et al. [46] recently propose a dense pixel-wise estimation method. Apart from generating 2D heat-maps, this method estimates 3D offsets of hand joints for each pixel of the 2D image. However, this method suffers from two limitations. First, as it regresses pixel-wise 3D estimations from 2D images, the proposed method may not fully exploit the 3D spatial information in depth images. Second, generating 3D estimations for background pixels of the 2D image may distract the deep neural network from learning effective features in the hand region.

To tackle these problems, we aim at regressing point-wise estimations directly from 3D point cloud, since the depth image is intrinsically composed of a set of 3D points on the visible object surface. We take advantages of PointNet [25, 27] to learn features directly from 3D point cloud. Compared with [46], our method can better utilize the 3D spatial information in the depth image in an efficient way and concentrate on learning effective features of the hand point cloud in a natural way, since both the input and the output of our network directly take the form of hand point cloud. In addition, this point-to-point regression scheme also allows us to expand the single hierarchical PointNet module [27] to a stacked network architecture as in [18] to further improve the estimation accuracy.

As illustrated in Figure 1, we propose a point-to-point regression method for 3D hand pose estimation in single depth images. Hand is first segmented from the depth image and is converted to a set of 3D points. The downsampled and normalized 3D points are then fed into a hierarchical PointNet [27] with two-stacked network architecture. The outputs of the network are heat-maps and unit vector fields on the 3D point cloud, reflecting the closeness and directions from 3D points to the target hand joints, respectively. Point-wise offsets to hand joints are inferred from the network outputs and are used to vote for 3D hand joint locations. With post-processing steps to alleviate other limitations, the estimation accuracy is further improved.

Our main contributions are summarized as follows:

- We propose to directly take the 3D point cloud as network input and generate heat-maps as well as unit vector fields on the input point cloud, which reflect the per-point closeness and directions to hand joints, respectively. With such a point-to-point regression network, our method is able to better utilize the 3D spatial information in the depth image and capture local structure of the 3D point cloud for accurate 3D hand pose estimation.
- We propose to apply the stacked network architecture [18] to the hierarchical PointNet [27] for point-to-point regression, which is the first stacked PointNet architecture, to our best knowledge. Similar to [18], the stacked PointNet architecture, feeding the output of one module as input into the

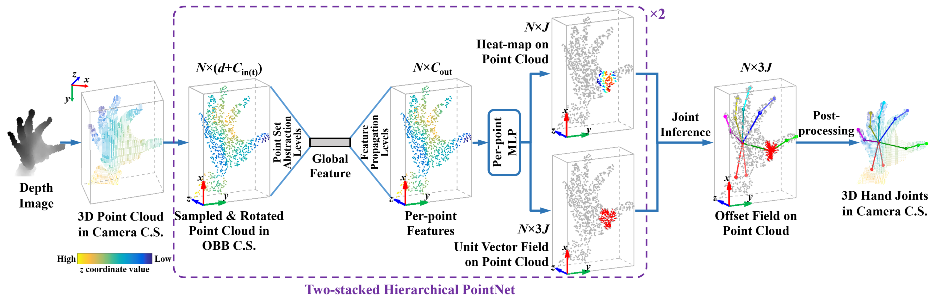
Depth Image
3D Point Cloud in Camera C.S.
High
Low
z coordinate value
$N\times(d+C_{in(t)})$
Sampled & Rotated Point Cloud in OBB C.S.
Point Set Abstraction Levels
Global Feature
Feature Propagation Levels
$N\times C_{out}$
Per-point Features
Per-point MLP
$N\times J$
Heat-map on Point Cloud
$N\times 3J$
Unit Vector Field on Point Cloud
×2
Two-stacked Hierarchical PointNet
Joint Inference
$N\times 3J$
Offset Field on Point Cloud
Post-processing
3D Hand Joints in Camera C.S.

Fig. 1. Overview of our proposed point-to-point regression method for 3D hand pose estimation from single depth images. We propose to directly take N sampled and normalized 3D hand points as network input and output a set of heat-maps as well as unit vector fields on the input point cloud, reflecting the closeness and directions from input points to J hand joints, respectively. From the network outputs, we can infer point-wise offsets to hand joints and estimate the 3D hand pose with post-processing. We apply the hierarchical PointNet [27] with two-stacked network architecture which feeds the output of one module as input to the next. For illustration purpose, we only visualize the heat-map, unit vector field and offset field of one hand joint. 'C.S.' stands for coordinate system; 'MLP' stands for multi-layer perceptron network.

next, allows repeated bottom-up and top-down inference on 3D point cloud and is able to boost the estimation accuracy in our experiments.

- We analyze the limitations of our point-to-point regression method and propose to use results of direct regression method as the alternative when the divergence among candidate estimations of point-to-point regression method is too large. Experiments show that the direct regression method is complementary with the point-to-point regression method and their combination can further improve the estimation accuracy.

We conduct extensive experiments on three challenging hand pose datasets: NYU dataset [43], ICVL dataset [36] and MSRA datasets [35]. Experimental results on these three datasets show that our proposed point-to-point regression method can achieve superior performance with runtime speed of 41.8fps and network model size of 17.2MB.

2 Related Work

Hand Pose Estimation The methods for 3D hand pose estimation from depth images can be classified into three categories: generative methods, discriminative methods and hybrid methods. Generative methods aim at fitting a deformable 3D hand model to the 3D point cloud converted from the input depth image [23, 1, 44, 14, 31, 40, 28]. Discriminative methods use training data to learn a mapping from a representation of the input depth image to a representation of the 3D hand pose [13, 51, 36, 35, 11, 10, 12, 19, 4, 5]. Hybrid methods combine a discriminative model learned from training data for pose estimation with a generative hand model for pose optimization [38, 43, 22, 45, 53, 32, 37].

Our work is related to research on 3D hand pose estimation with deep neural networks-based approaches [11, 22, 45, 10, 53, 12, 19, 4, 5, 2]. Tompson et al. [43] first propose to apply CNNs in 3D hand pose estimation. They use CNNs to generate heat-maps representing the 2D probability distributions of hand joints in the depth image, and recover 3D hand pose from estimated heat-maps and corresponding depth values using model-based inverse kinematics. Ge et al. [10] solve the problem of lacking 3D information in 2D heat-maps [43] by projecting the depth image onto multiple views and estimating 3D hand pose from multi-view heat-maps. Oberweger et al. [21, 19] instead directly regress 3D coordinates of hand joints or a lower dimensional embedding of 3D hand pose from depth images. They also propose a feedback loop [22] to iteratively refine the 3D hand pose. Zhou et al. [56] propose to directly regress hand model parameters from depth images. Ge et al. [11] encode the hand depth images as 3D volumes and use 3D CNNs to directly regress 3D hand pose from 3D volumes. Guo et al. [12] propose a region ensemble network that directly regresses 3D hand pose from depth images. Chen et al. [4] improve [12] through iterative refinement. Although many 3D hand pose estimation methods directly regress 3D hand pose, Wan et al. [46] recently propose a dense pixel-wise estimation method that applies an hourglass network to generate 2D and 3D heat-maps as well as 3D unit vector fields, from which the 3D hand joint locations can be inferred. Our method is inspired by this work [46], but is essentially different from it. Firstly, the network proposed in [46] takes 2D images as input, while our method takes 3D point cloud as the network input, thus is able to better utilize 3D spatial information in the depth image. Secondly, the network proposed in [46] outputs estimations for each pixel in the original image which may contain large useless background regions, while our proposed point-to-point regression network outputs estimations for each point in the hand point cloud, thus is able to concentrate on learning effective features from the hand point cloud instead of background regions.

3D Deep Learning 3D data usually are not suitable to be directly processed by conventional CNNs that work on 2D images. Methods in [10, 34, 26, 3] project 3D points into 2D images on multiple views and process them with multi-view CNNs. Methods in [11, 26, 50, 16, 33] rasterize 3D points into 3D voxels and apply 3D CNNs to extract features. But the time and space complexities of 3D CNNs are high. Octree-based 3D CNNs [29, 48] are then proposed for efficient computation on 3D volumes with high resolution, but still suffer from voluminous input data.

PointNet [25, 27] is a recently proposed method that directly takes an unordered point set as input and is able to learn features on the point set. In the basic PointNet [25], each input point is mapped into a feature vector via multi-layer perceptron networks (MLP), of which the weights are shared across all the input points. Then, a vector max operator aggregates per-point features into a global feature that is invariant to different permutations of input points. The extracted global feature and per-point features can be used for various tasks. The basic PointNet [25] cannot capture local structures of the point cloud. To tackle this issue, a hierarchical PointNet [27] is proposed to extract local features in a hierarchical way. We refer readers to [27] for the details of hierarchical PointNet. Deep Kd-networks [15], similar to PointNet, directly consumes point cloud by adopting a Kd-tree structure. Although these methods have shown promising performance on 3D classification and segmentation tasks, none of them has been applied to 3D hand pose estimation in a point-to-point regression manner.

3 Methodology

Our proposed method aims at estimating 3D hand pose from single depth images. The input is a depth image containing a hand, and the output is a set of 3D hand joint locations $\mathbf{\Phi}^{cam} = \left\{\boldsymbol{\phi}_j^{cam}\right\}_{j=1}^{J} \in \mathbf{\Lambda}$ in the camera Coordinate System (C.S.), where J is the number of hand joints, $\mathbf{\Lambda}$ is the $3 \times J$ dimensional hand joint space.

3.1 Point Cloud Preprocessing

The hand depth image is first converted to a set of 3D points using the depth camera's intrinsic parameters. The 3D point set is then downsampled to N points. To make our method robust to various hand orientations, we create an oriented bounding box (OBB) from the 3D point cloud and transform the 3D points into the OBB C.S., as shown in Fig. 1. The coordinates of 3D points are normalized between -0.5 and 0.5 by subtracting the centroid of point cloud and dividing by L_{obb}, which is the maximum edge length of OBB. We denote the downsampled and normalized 3D point set in OBB C.S. as $\mathcal{P}^{obb} = \left\{\boldsymbol{p}_i^{obb}\right\}_{i=1}^{N}$. In our implementation, we set the number of sampled points N as 1024. Since we process the point set in OBB C.S., we will omit the superscript 'obb' in symbols of points and joint locations in the following sections for simplicity.

3.2 Point Cloud based Representation for 3D Hand Pose

Most existing CNN-based methods for 3D hand pose estimation directly regress 3D coordinates of hand joints [11, 22, 45, 12, 4] or hand pose parameters [19, 5, 21, 56]. In contrast to direct regression approaches that require to learn a highly non-linear mapping, our method aims at generating point-wise estimations of hand joint locations from the point cloud, which is able to better utilize the local evidence. The point-wise estimations can be defined as the offsets from points to hand joint locations. However, estimating offsets for all points in the point set is unnecessary and may make the per-point votes noisy. Thus, we only estimate offsets for the neighboring points of the hand joint, as shown in Fig. 2. We define the element in the target offset fields $\boldsymbol{V}$ for point $\boldsymbol{p}_i \, (i = 1, \cdots, N)$ and ground truth hand joint location $\boldsymbol{\phi}_j^* \, (j = 1, \cdots, J)$ as:

$$\boldsymbol{V}\left(\boldsymbol{p}_i, \boldsymbol{\phi}_j^*\right) = \begin{cases} \boldsymbol{\phi}_j^* - \boldsymbol{p}_i & \boldsymbol{p}_i \in \mathcal{P}_K\left(\boldsymbol{\phi}_j^*\right) \text{ and } \left\|\boldsymbol{\phi}_j^* - \boldsymbol{p}_i\right\| \leq r, \\ \mathbf{0} & \text{otherwise;} \end{cases} \tag{1}$$

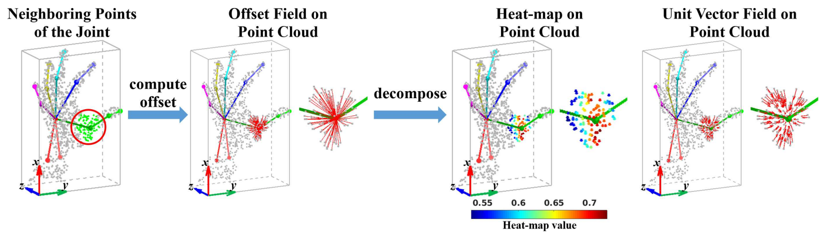
Neighboring Points of the Joint
compute offset
Offset Field on Point Cloud
decompose
Heat-map on Point Cloud
Unit Vector Field on Point Cloud
0.55
0.6
0.65
0.7
Heat-map value
x
y
z

Fig. 2. An illustration of the ground truth of the point cloud based representation for 3D hand pose. We visualize the neighboring points, offset field, heat-map and unit vector field on 3D point cloud for the root joint of the thumb finger. For illustration propose, we enlarge the region of neighboring points of the hand joint location on the right of each complete point cloud.

where $\mathcal{P}_K\left(\boldsymbol{\phi}_j^*\right)$ is a set of K nearest neighboring points (KNN) of the ground truth hand joint location $\boldsymbol{\phi}_j^*$ in the point set $\mathcal{P}^{obb}$; r is the maximum radius of ball for nearest neighbor search; in our implementation, we set K as 64 and r as $80mm/L_{obb}$. We combine KNN with ball query for nearest neighbor search in order to guarantee that both the number of neighboring points and the scale of neighboring region are controllable.

However, it is difficult to train a neural network that directly generates the offset field due to the large variance of offsets. Similar to [46], we decompose the target offset fields $\boldsymbol{V}$ into heat-maps H reflecting per-point closeness to hand joint locations:

$$H\left(\boldsymbol{p}_i, \boldsymbol{\phi}_j^*\right) = \begin{cases} 1 - \left\|\boldsymbol{\phi}_j^* - \boldsymbol{p}_i\right\|/r & \boldsymbol{p}_i \in \mathcal{P}_K\left(\boldsymbol{\phi}_j^*\right) \text{ and } \left\|\boldsymbol{\phi}_j^* - \boldsymbol{p}_i\right\| \le r, \\ 0 & \text{otherwise;} \end{cases} \quad (2)$$

and unit vector fields $\boldsymbol{U}$ reflecting per-point directions to hand joint locations:

$$\boldsymbol{U}\left(\boldsymbol{p}_i, \boldsymbol{\phi}_j^*\right) = \begin{cases} \left(\boldsymbol{\phi}_j^* - \boldsymbol{p}_i\right)/\left\|\boldsymbol{\phi}_j^* - \boldsymbol{p}_i\right\| & \boldsymbol{p}_i \in \mathcal{P}_K\left(\boldsymbol{\phi}_j^*\right) \text{ and } \left\|\boldsymbol{\phi}_j^* - \boldsymbol{p}_i\right\| \le r, \\ \boldsymbol{0} & \text{otherwise.} \end{cases} \quad (3)$$

Different from [46] that generates heat-maps and unit vector fields on 2D images, our proposed method generates heat-maps and unit vector fields on the 3D point cloud, as shown in Fig. 2, which can better utilize the 3D spatial information in the depth image. In addition, generating heat-maps and unit vector fields on 2D images with large blank background regions may distract the neural network from learning effective features in the hand region. Although this problem can be alleviated by multiplying a binary hand mask in the loss function, our method is able to concentrate on learning effective features of the hand point cloud in a natural way without using any mask, since the output heat-maps and unit vector fields are represented on the hand point cloud.

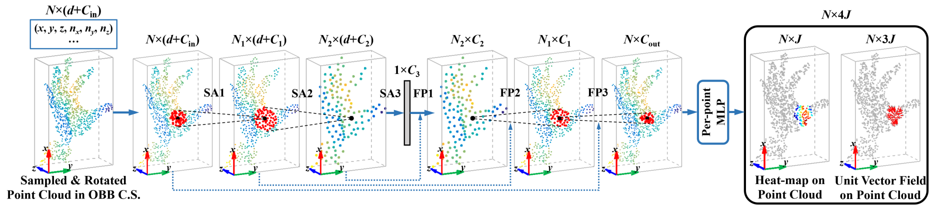

Fig. 3. An illustration of a single network module which is based on the hierarchical PointNet [27]. Here, 'SA' stands for point set abstraction layers; 'FP' stands for feature propagation layers; 'MLP' stands for multi-layer perceptron network. The dotted shortcuts denote skip links for feature concatenation.

3.3 Network Architecture

In this work, we exploit the hierarchical PointNet [27] for learning heat-maps and unit vector fields on 3D point cloud. Different from the hierarchical PointNet for point set segmentation adopted in [27], our proposed point-to-point regression network has a two-stacked network architecture in order to better capture the 3D spatial information in the 3D point cloud.

We first describe the network architecture of a single hierarchical PointNet module. As illustrated in Fig. 3, the input of the network is a set of d-dim coordinates with C_{in}-dim input features, *i.e.*, 3D surface normals that are approximated by fitting a local plane for the nearest neighbors of the query point in the point cloud ($d = 3$ and $C_{in} = 3$ in this work). Similar to the network architecture for set segmentation proposed in [27], a single module of our network extracts a global feature vector from point cloud using three set abstraction levels and propagates the global feature to point features for original points using three feature propagation levels, as shown in Fig. 3. In the feature propagation level, we use nearest neighbors of the interpolation point in N_l points to interpolate features for N_{l-1} points [27]. The interpolated C_l-dim features of N_{l-1} points are concatenated with the corresponding point features in the set abstraction level and are mapped to C_{l-1}-dim features using per-point MLP, of which the weights are shared across all the points ($l = 1, 2, 3; N_0 = N, C_0 = C_{out}, N_3 = 1$). The heat-map and the unit vector field are generated from the point features for the original point set using per-point MLP. In our implementation, we set $N = 1024$, $N_1 = 512$, $N_2 = 128$, $C_1 = 128$, $C_2 = 256$ and $C_{out} = 128$.

Inspired by the stacked hourglass networks for human pose estimation [18], we stack two hierarchical PointNet modules end-to-end to boost the performance of the network. The two hierarchical PointNet modules have the same network architecture and the same hyper-parameters, except for the hyper-parameter in the input layer. As shown in Fig. 4, the output heat-map and unit vector field of the first module are concatenated with the input and output point features of the first module as the input into the second hierarchical PointNet module. For real-time consideration, we only stack two hierarchical PointNet modules.

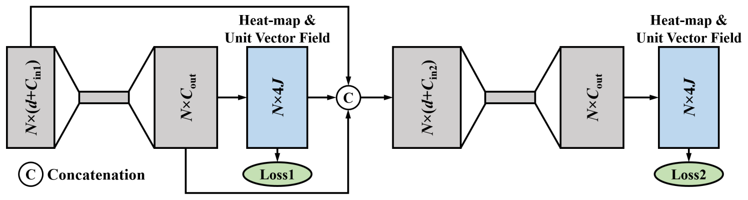

Fig. 4. An illustration of the two-stacked hierarchical PointNet architecture with intermediate supervision. The input feature dimension of the 2nd network module is $C_{in2} = C_{in1} + C_{out} + 4J$.

We apply intermediate supervision when training the two-stacked hierarchical PointNet. The loss function for each training sample is defined as:

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i=1}^{N} \left[\left(\hat{H}_{ij}^{(t)} - H\left(\boldsymbol{p}_i, \boldsymbol{\phi}_j^*\right) \right)^2 + \left\| \hat{\boldsymbol{U}}_{ij}^{(t)} - \boldsymbol{U}\left(\boldsymbol{p}_i, \boldsymbol{\phi}_j^*\right) \right\|^2 \right], \tag{4}$$

where T is the number of stacked network modules, in this work $T = 2$; $\hat{H}_{ij}^{(t)}$ and $\hat{\boldsymbol{U}}_{ij}^{(t)}$ are elements in the heat-maps and unit vector fields estimated by the t-th network module, respectively; $H\left(\boldsymbol{p}_i, \boldsymbol{\phi}_j^*\right)$ and $\boldsymbol{U}\left(\boldsymbol{p}_i, \boldsymbol{\phi}_j^*\right)$ are elements in the ground truth heat-maps and ground truth unit vector fields defined in Eq. 2 and Eq. 3, respectively.

3.4 Hand Pose Inference

During testing, we infer the 3D hand pose from the heat-maps $\hat{H}$ and the unit vector fields $\hat{\boldsymbol{U}}$ estimated by the last hierarchical PointNet module. According to the definition of offset fields, heat-maps and unit vector fields in Eq. 1-3, we can infer the offset vector $\hat{\boldsymbol{V}}_{ij}$ from point $\boldsymbol{p}_i$ to joint $\hat{\boldsymbol{\phi}}_j$ as:

$$\hat{\boldsymbol{V}}_{ij} = r \cdot \left(1 - \hat{H}_{ij}\right) \cdot \hat{\boldsymbol{U}}_{ij}. \tag{5}$$

According to Eq. 1, only the offset vectors for the neighboring points of the hand joint are used for hand pose inference, which can be found from the estimated heat-map reflecting the closeness of points to the hand joint. We denote the estimated heat-map for the j-th hand joint as $\hat{H}_j$ that is the j-th column of $\hat{H}$. We determine the neighboring points of the j-th hand joint as the points corresponding to the largest M values of the heat-map $\hat{H}_j$. The indices of these points in the point set are denoted as $\{i_m\}_{m=1}^{M}$. The hand joint location $\hat{\boldsymbol{\phi}}_j$ can be simply inferred from the corresponding offset vectors $\hat{\boldsymbol{V}}_{i_m j}$ and 3D points $\boldsymbol{p}_{i_m}$ $(m = 1, \cdots, M)$ using weighted average:

$$\hat{\boldsymbol{\phi}}_j = \sum\nolimits_{m=1}^{M} w_m \left(\hat{\boldsymbol{V}}_{i_m j} + \boldsymbol{p}_{i_m}\right) \Big/ \sum\nolimits_{m=1}^{M} w_m, \tag{6}$$

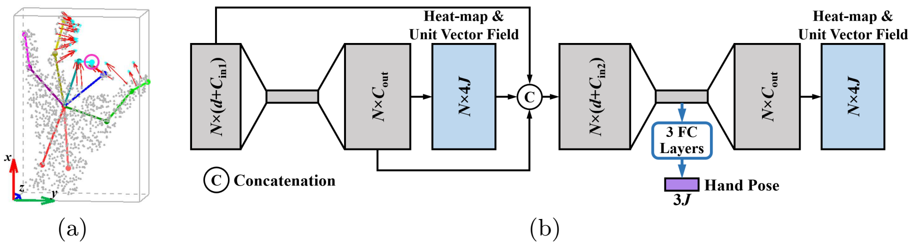

Fig. 5. (a) A failure case in which the candidate estimations of the middle fingertip can not converge to a small local region in 3D space due to missing depth data near the hand joint. The ground truth hand joint locations are plotted in this figure. (b) An illustration of the two-stacked hierarchical PointNet architecture in which we add three fully-connected layers to directly regress the 3D coordinates of hand joints from the global feature extracted by the second hierarchical PointNet module.

where w_m is the weight of the candidate estimation. In our implementation, we set the weight w_m as the corresponding heat-map value $\hat{H}_{i_m j}$, and set M as 25.

3.5 Post-processing

There are two issues in our point-to-point regression method. The first issue is that the estimation is unreliable when the divergence of the M candidate estimations are large in 3D space, as shown in Fig. 5(a). This is usually caused by missing depth data near the hand joint. The second issue is that there is no explicit constraint on the estimated 3D hand pose, although the neural network may learn joint constraints in the output heat-maps and unit vector fields.

To tackle the first issue, when the divergence of the M candidate estimations is larger than a threshold, we replace the estimation result with the result of the direct regression method that directly regresses 3D coordinates of hand joints, since the direct regression method does not have this issue. In order to save the inference time, instead of training a separate PointNet for direct hand pose regression, we add three fully-connected layers for direct hand pose regression to the pre-trained two-stacked hierarchical PointNet, as shown in Fig. 5(b). The three fully-connected layers are trained to directly regress the 3D coordinates of hand joints from the features extracted by the second hierarchical PointNet module. The divergence of the M candidate estimations is defined as the sum of standard deviations of x, y and z coordinates of candidate estimations. In our implementation, we set the divergence threshold as $7.5mm/L_{obb}$. Experimental results in Section 4.1 will show that although only a small portion of the hand joint estimations requires to be replaced by the direct regression results, this replacement strategy can improve the estimation accuracy to some extent.

To tackle the second issue, we explicitly constrain the estimated 3D hand pose $\hat{\mathbf{\Phi}}$ on a lower dimensional space learned by principal component analysis (PCA). By performing PCA on the ground truth 3D joint locations in the training dataset, we can obtain the principal components $\boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_H]$ $(H < 3J)$

and the empirical mean $\boldsymbol{u}$. The constrained 3D hand pose can be calculated using the following formula:

$$\hat{\boldsymbol{\Phi}}_{cons} = \boldsymbol{E} \cdot \boldsymbol{E}^T \cdot \left(\hat{\boldsymbol{\Phi}} - \boldsymbol{u} \right) + \boldsymbol{u}. \tag{7}$$

In our implementation, we set the number of principle components H as 30. Experimental results in Section 4.1 will show that adding PCA constraint will improve the accuracy slightly, which shows that the neural network may have already learned joint constraints in the output heat-maps and unit vector fields.

Finally, the estimated 3D hand joint locations in the normalized OBB C.S. are transformed back to joint locations in the camera C.S. $\hat{\boldsymbol{\Phi}}^{cam}$.

4 Experiments

We evaluate our proposed method on three public hand pose datasets: NYU dataset [43], ICVL dataset [36] and MSRA dataset [35]. NYU dataset [43] contains 72,757 frames for training samples and 8,252 frames for testing. The ground truth of each frame contains 3D locations of 36 hand joints. Following previous work in [43, 11, 22], we estimate and evaluate on a subset of 14 hand joints. Since the frames in this dataset are original depth images containing human body and background, we use a single hourglass network [18] to detect 2D hand joint locations and use the corresponding depth information for hand segmentation. We augment the training data with random arm lengths due to various lengths of hand arm in the segmented images. ICVL dataset [36] contains 22,059 frames for training and 1,596 frames for testing. The ground truth of each frame contains 3D locations of 16 hand joints. We use the same method as that used on NYU dataset for hand segmentation. The training data is randomly augmented with various arm lengths and stretch factors. MSRA dataset [35] contains nine subjects, each subject contains 17 hand gestures and each hand gesture contains about 500 frames with segmented hand depth image. The ground truth of each frame contains 3D locations of 21 hand joints. In the experiments, we train on eight subjects and test on the remaining one. This is repeated nine times for all subjects. We do not perform any data augmentation on this dataset.

We adopt two metrics to evaluate the performance of 3D hand pose estimation methods. The first metric is the per-joint mean error distance over all test frames as well as the overall mean error distance for all joints on all test frames. The second metric is the proportion of good frames in which the worst joint error is below a threshold [39]. This metric is more strict.

We train and evaluate our proposed deep neural network models on a workstation with two Intel Core i7 5930K, 64GB of RAM and an Nvidia TITAN Xp GPU. The deep neural network models are implemented within the PyTorch framework. When training the deep neural network models, we use Adam [14] optimizer with initial learning rate 0.001, batch size 32, momentum 0.5 and weight decay 0.0005. The learning rate is divided by 10 after 30 epochs. The training is stopped after 60 epochs to prevent overfitting.

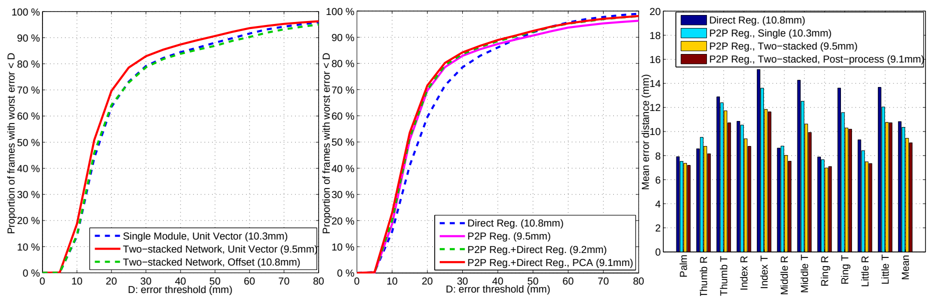

Fig. 6. Self-comparison of different methods on NYU dataset [43]. **Left**: the impacts of the stacked network architecture and different network outputs on the proportion of good frames. **Middle**: the impacts of our point-to-point regression method and post-processing methods on the proportion of good frames. We use two-stacked network for point-to-point regression in this figure. **Right**: the impact of point-to-point regression method, stacked network architecture and post-processing methods on the per-joint mean error distance (R: root, T: tip). 'P2P Reg.' stands for point-to-point regression. The overall mean error distances are shown in parentheses.

# Candidate Estimations M	5	15	25	35	45	55
Unweighted Average	9.50mm	9.47mm	9.48mm	9.57mm	9.70mm	9.84mm
Weighted Average	9.50mm	9.47mm	9.46mm	9.53mm	9.61mm	9.71mm

Table 1. The impacts of the number of candidate estimations M and weighted average on the overall mean error distance on NYU dataset [43].

4.1 Self-comparisons

We first evaluate the impact of the stacked network architecture for hierarchical PointNet. As shown in Fig. 6 (left and right), the two-stacked network evidently performs better than the single network module, which indicates the importance of the stacked network architecture on our point-to-point regression method.

We also evaluate the impact of different network outputs. In our method, we train the network to output heat-maps and unit vector fields for hand joints, then use them to recover the offset fields, as described in Section 3. In this experiment, we compare our method with a baseline method in which a network is trained to generate offset fields instead of unit vector fields. The network also outputs heat-maps which are only used to find neighboring points of hand joints. As shown in Fig. 6 (left), when adopting the two-stacked network architecture, the network generating unit vector fields performs better than the network generating offset fields. This result shows that the network regressing unit vectors of offsets may be easier to learn than the network regressing offset vectors, since the variance of the offset vectors is larger than the unit vectors.

To evaluate our proposed point-to-point regression method, we compare our method with the direct regression method. In this experiment, we use a hierarchical PointNet [27] with three set abstraction levels and three full-connected levels to directly regress the 3D coordinates of hand joints. As shown in Fig. 6 (middle), our point-to-point regression method outperforms the direct regression method when the error threshold is smaller than 45mm. But when the error threshold is larger than 45mm, our point-to-point regression method performs worse than the direct regression method. This may be caused by the large divergence of the candidate estimations in some results, as described in Section 3.5. By combining the point-to-point method with the direct regression method as described in Section 3.5, the estimation accuracy can be further improved, as shown in Fig. 6 (middle). Furthermore, the performance of the combination method is superior to or on par with the direct regression method over all the error thresholds. In this experiment, only 7.9% of joint locations estimated by point-to-point regression method are replaced by the results of direct regression method, which indicates that the estimation results are dominated by the point-to-point regression method, and the direct regression method is complementary with the point-to-point regression method. In addition, adding the PCA constraint can further improve the estimation accuracy slightly.

We further study the influence of the number of candidate estimations M used in Eq. 7 and the weighted average on the overall mean error distance. As shown in Table 1, the mean error distance is the smallest when the number of candidate estimations M is between 15 and 25. When M is larger than 25, the mean error distance will become larger. In addition, when M is smaller than 25, the weighted average will not improve the mean error distance. But when M becomes larger, the improvement of the weighted average on the mean error distance is more and more evident. Thus, the weighted average is able to make the estimation more robust to noisy candidate estimations. We set M as 25 and use weighted average with post-processing in the following experiments.

4.2 Comparisons with State-of-the-arts

We compare our proposed point-to-point regression method with 16 state-of-the-art methods: latent random forest (LRF) [36], hierarchical regression with random forest (RDF, Hierarchical) [35], local surface normal based random forest (LSN) [47], collaborative filtering [6], 2D heat-map regression using 2D CNNs (Heat-map) [43], feedback loop based 2D CNNs (Feedback Loop) [22], hand model parameter regression using 2D CNNs (DeepModel) [56], Lie group based 2D CNNs (Lie-X) [52], improved direct regression with a pose prior using 2D CNNs (DeepPrior++) [19], hallucinating heat distribution using 2D CNNs (Hallucination Heat) [5], multi-view CNNs [10], 3D CNNs [11], crossing nets using deep generative models (Crossing Nets) [45], region ensemble network (REN) [12], pose guided structured REN (Pose-REN) [4] and dense 3D regression using 2D CNNs (DenseReg) [46]. We evaluate the proportion of good frames over different error thresholds and the per-joint mean error distances as well as the overall mean error distance on NYU [43], ICVL [36] and MSRA [35] datasets, as presented in Fig. 7 and Fig. 8, respectively.

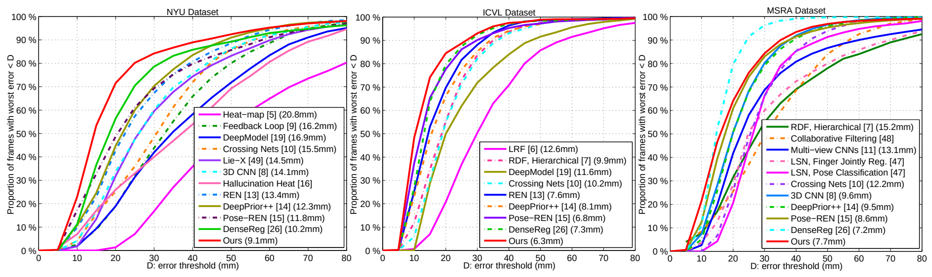

Fig. 7. Comparison with state-of-the-art methods on NYU [43] (left), ICVL [36] (middle) and MSRA [35] (right) datasets. The proportions of good frames and the overall mean error distances (in parentheses) are presented in this figure.

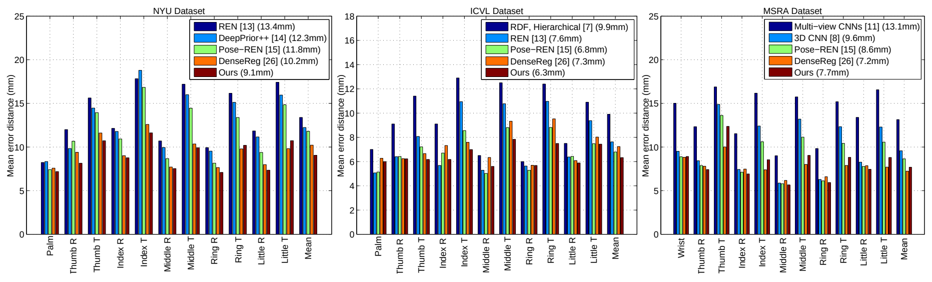

Fig. 8. Comparison with state-of-the-art methods on NYU [43] (left), ICVL [36] (middle) and MSRA [35] (right) datasets. The per-joint mean error distances and the overall mean error distances are presented in this figure (R: root, T: tip).

As can be seen in Fig. 7 and Fig. 8, our method can achieve superior performance on these three datasets. On NYU [43] and ICVL [36] datasets, our method outperforms other methods over almost all the error thresholds and achieves the smallest overall mean error distances on these two datasets. Specifically, on NYU dataset [43], when the error threshold is between 15mm and 20mm, the proportions of good frames of our method is about 15% better than DenseReg [46] and 20% batter than Pose-REN [4]; on ICVL dataset [36], when the error threshold is between 10mm and 15mm, the proportions of good frames of our method is more than 10% better than DenseReg [46] and Pose-REN [4] methods. On MSRA dataset [35], our method outperforms other methods over almost all the error thresholds, except for the DenseReg [46] method. Although our method is about 10% better than DenseReg [46] when the error threshold is 10mm and the overall mean error distance of our method is only 0.5mm worse than that of DenseReg [46], our method is worse than the DenseReg [46] method when the error threshold is larger 15mm. As mentioned in [20] and shown in the qualitative results, some of the 3D hand joint annotations in MSRA dataset [35] exhibit significant errors, which may make the evaluation on this dataset less meaningful and may limit the learning ability of our deep neural network.

In addition, we present some qualitative results for NYU [43], ICVL [36] and MSRA [35] datasets in the supplementary material.

4.3 Runtime and Model Size

The runtime of our method is 23.9ms per frame in average, including 8.2ms for point sampling and surface normal calculation, 15.1ms for the two-stacked hierarchical PointNet forward propagation, 0.6ms for hand pose inference and post-processing. Thus, our method runs in real-time at about 41.8fps.

In addition, the model size our network is 17.2MB, including 11.1MB for the point-to-point regression network which is a two-stacked hierarchical PointNet and 6.1MB for the additional direct regression module which consists of three fully-connected layers. Compared with the model size of the 3D CNNs proposed in [11] which is about 420MB, our model size is smaller.

5 Conclusion

In this paper, we propose a novel approach that directly takes the 3D point cloud of hand as network input and outputs heat-maps as well as unit vector fields on the point cloud, reflecting the per-point closeness and directions to hand joints. We infer 3D hand joint locations from the estimated heat-maps and unit vector fields using weighted fusion. Similar to the stacked hourglass network [18], we apply the stacked network architecture for the hierarchical PointNet [27], which allows repeated bottom-up and top-down inference on point cloud and is able to further boost the performance. Our proposed point-to-point regression method can also be easily combined with direct regression method to achieve more robust performance. Experimental results on three challenging hand pose datasets show that our method achieves superior accuracy performance in real-time.

Acknowledgment

This research is 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. This work is also supported in part by start-up grants from University at Buffalo and a gift grant from Snap Inc.

References

1. Ballan, L., Taneja, A., Gall, J., Gool, L.V., Pollefeys, M.: Motion capture of hands in action using discriminative salient points. In: ECCV (2012)

2. Cai, Y., Ge, L., Cai, J., Yuan, J.: Weakly-supervised 3d hand pose estimation from monocular rgb images. In: ECCV (2018)
3. Cao, Z., Huang, Q., Ramani, K.: 3d object classification via spherical projections. In: 3DV (2017)
4. Chen, X., Wang, G., Guo, H., Zhang, C.: Pose guided structured region ensemble network for cascaded hand pose estimation. arXiv preprint arXiv:1708.03416 (2017)
5. Choi, C., Kim, S., Ramani, K.: Learning hand articulations by hallucinating heat distribution. In: ICCV (2017)
6. Choi, C., Sinha, A., Hee Choi, J., Jang, S., Ramani, K.: A collaborative filtering approach to real-time hand pose estimation. In: ICCV (2015)
7. Ge, L., Liang, H., Yuan, J., Thalmann, D.: Real-time 3d hand pose estimation with 3d convolutional neural networks. IEEE Trans. Pattern Anal. Mach. Intell. pp. 1–15 (2018)
8. Ge, L., Liang, H., Yuan, J., Thalmann, D.: Robust 3d hand pose estimation from single depth images using multi-view cnns. IEEE Trans. Image Processing **27**(9), 4422–4436 (2018)
9. Ge, L., Cai, Y., Weng, J., Yuan, J.: Hand pointnet: 3D hand pose estimation using point sets. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog. pp. 8417–8426 (2018)
10. Ge, L., Liang, H., Yuan, J., Thalmann, D.: Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs. In: CVPR (2016)
11. Ge, L., Liang, H., Yuan, J., Thalmann, D.: 3D convolutional neural networks for efficient and robust hand pose estimation from single depth images. In: CVPR (2017)
12. Guo, H., Wang, G., Chen, X., Zhang, C., Qiao, F., Yang, H.: Region ensemble network: Improving convolutional network for hand pose estimation. In: ICIP (2017)
13. Keskin, C., Kra, F., Kara, Y., Akarun, L.: Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In: ECCV (2012)
14. Khamis, S., Taylor, J., Shotton, J., Keskin, C., Izadi, S., Fitzgibbon, A.: Learning an efficient model of hand shape variation from depth images. In: CVPR (2015)
15. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: ICCV (2017)
16. Maturana, D., Scherer, S.: Voxnet: A 3D convolutional neural network for real-time object recognition. In: IROS (2015)
17. Moon, G., Chang, J.Y., Lee, K.M.: V2V-PoseNet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In: CVPR (2018)
18. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: ECCV (2016)
19. Oberweger, M., Lepetit, V.: Deepprior++: Improving fast and accurate 3d hand pose estimation. In: ICCV Workshop (2017)
20. Oberweger, M., Riegler, G., Wohlhart, P., Lepetit, V.: Efficiently creating 3d training data for fine hand pose estimation. In: CVPR (2016)
21. Oberweger, M., Wohlhart, P., Lepetit, V.: Hands deep in deep learning for hand pose estimation. In: CVWW (2015)
22. Oberweger, M., Wohlhart, P., Lepetit, V.: Training a feedback loop for hand pose estimation. In: ICCV (2015)
23. Oikonomidis, I., Kyriazis, N., Argyros, A.: Efficient model-based 3D tracking of hand articulations using Kinect. In: BMVC (2011)
24. Pavlakos, G., Zhou, X., Derpanis, K.G., Daniilidis, K.: Coarse-to-fine volumetric prediction for single-image 3d human pose. In: CVPR (2017)

25. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: CVPR (2017)
26. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: CVPR (2016)
27. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: NIPS (2017)
28. Remelli, E., Tkach, A., Tagliasacchi, A., Pauly, M.: Low-dimensionality calibration through local anisotropic scaling for robust hand model personalization. In: ICCV (2017)
29. Riegler, G., Ulusoy, A.O., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: CVPR (2017)
30. Rogez, G., Weinzaepfel, P., Schmid, C.: Lcr-net: Localization-classification-regression for human pose. In: CVPR (2017)
31. Romero, J., Tzionas, D., Black, M.J.: Embodied hands: modeling and capturing hands and bodies together. ACM Transactions on Graphics (TOG) **36**(6), 245 (2017)
32. Sharp, T., Keskin, C., Robertson, D., Taylor, J., Shotton, J., Kim, D., Rhemann, C., Leichter, I., Vinnikov, A., Wei, Y., Freedman, D., Kohli, P., Krupka, E., Fitzgibbon, A., Izadi, S.: Accurate, robust, and flexible real-time hand tracking. In: CHI (2015)
33. Song, S., Xiao, J.: Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In: CVPR (2016)
34. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3D shape recognition. In: ICCV (2015)
35. Sun, X., Wei, Y., Liang, S., Tang, X., Sun, J.: Cascaded hand pose regression. In: CVPR (2015)
36. Tang, D., Chang, H.J., Tejani, A., Kim, T.K.: Latent regression forest: Structured estimation of 3D articulated hand posture. In: CVPR (2014)
37. Tang, D., Taylor, J., Kohli, P., Keskin, C., Kim, T.K., Shotton, J.: Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In: ICCV (2015)
38. Taylor, J., Bordeaux, L., Cashman, T., Corish, B., Keskin, C., Sharp, T., Soto, E., Sweeney, D., Valentin, J., Luff, B., Topalian, A., Wood, E., Khamis, S., Kohli, P., Izadi, S., Banks, R., Fitzgibbon, A., Shotton, J.: Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. ACM Transactions on Graphics **35**(4), 143 (2016)
39. Taylor, J., Shotton, J., Sharp, T., Fitzgibbon, A.: The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In: CVPR (2012)
40. Tkach, A., Tagliasacchi, A., Remelli, E., Pauly, M., Fitzgibbon, A.: Online generative model personalization for hand tracking. ACM Transactions on Graphics (TOG) **36**(6), 243 (2017)
41. Tome, D., Russell, C., Agapito, L.: Lifting from the deep: Convolutional 3d pose estimation from a single image. In: CVPR (2017)
42. Tompson, J., Jain, A., LeCun, Y., Bregler, C.: Joint training of a convolutional network and a graphical model for human pose estimation. In: NIPS (2014)
43. Tompson, J., Stein, M., Lecun, Y., Perlin, K.: Real-time continuous pose recovery of human hands using convolutional networks. ACM Transactions on Graphics **33**(5), 169 (2014)
44. Tzionas, D., Ballan, L., Srikantha, A., Aponte, P., Pollefeys, M., Gall, J.: Capturing hands in action using discriminative salient points and physics simulation. International Journal of Computer Vision **118**(2), 172–193 (2016)

45. Wan, C., Probst, T., Van Gool, L., Yao, A.: Crossing nets: Dual generative models with a shared latent space for hand pose estimation. In: CVPR (2017)
46. Wan, C., Probst, T., Van Gool, L., Yao, A.: Dense 3d regression for hand pose estimation pp. 5147–5156 (2018)
47. Wan, C., Yao, A., Van Gool, L.: Direction matters: hand pose estimation from local surface normals. In: ECCV (2016)
48. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions on Graphics (TOG) **36**(4), 72 (2017)
49. Wei, S.E., Ramakrishna, V., Kanade, T., Sheikh, Y.: Convolutional pose machines. In: CVPR (2016)
50. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D shapenets: A deep representation for volumetric shapes. In: CVPR (2015)
51. Xu, C., Cheng, L.: Efficient hand pose estimation from a single depth image. In: ICCV (2013)
52. Xu, C., Govindarajan, L.N., Zhang, Y., Cheng, L.: 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)
53. Ye, Q., Yuan, S., Kim, T.K.: Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation. In: ECCV (2016)
54. Yuan, S., Garcia-Hernando, G., Stenger, B., Moon, G., Chang, J.Y., Lee, K.M., Molchanov, P., Kautz, J., Honari, S., Ge, L., Yuan, J., Chen, X., Wang, G., Yang, F., Akiyama, K., Wu, Y., Wan, Q., Madadi, M., Escalera, S., Li, S., Lee, D., Oikonomidis, I., Argyros, A., Kim, T.K.: Depth-based 3D hand pose estimation: From current achievements to future goals. In: CVPR (2018)
55. Yuan, S., Ye, Q., Stenger, B., Jain, S., Kim, T.K.: Bighand2.2m benchmark: Hand pose dataset and state of the art analysis. In: CVPR (2017)
56. Zhou, X., Wan, Q., Zhang, W., Xue, X., Wei, Y.: Model-based deep hand pose estimation. In: IJCAI (2016)