2021 IEEE/CVF International Conference on Computer Vision (ICCV) | 978-1-6654-2812-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCV48922.2021.01107

HandFoldingNet: A 3D Hand Pose Estimation Network Using Multiscale-Feature Guided Folding of a 2D Hand Skeleton

Wencan Cheng[1], Jae Hyun Park[1], Jong Hwan Ko[2*]
[1]Department of Artificial Intelligence, Sungkyunkwan University
[2]College of Information and Communication Engineering, Sungkyunkwan University
{cwc1260, xoxc4565, jhko}@skku.edu

Abstract

With increasing applications of 3D hand pose estimation in various human-computer interaction applications, convolution neural networks (CNNs) based estimation models have been actively explored. However, the existing models require complex architectures or redundant computational resources to trade with the acceptable accuracy. To tackle this limitation, this paper proposes HandFoldingNet, an accurate and efficient hand pose estimator that regresses the hand joint locations from the normalized 3D hand point cloud input. The proposed model utilizes a folding-based decoder that folds a given 2D hand skeleton into the corresponding joint coordinates. For higher estimation accuracy, folding is guided by multi-scale features, which include both global and joint-wise local features. Experimental results show that the proposed model outperforms the existing methods on three hand pose benchmark datasets with the lowest model parameter requirement. Code is available at https://github.com/cwc1260/HandFold.

1. Introduction

3D hand pose estimation aims to estimate joint locations from input hand images. Accurate and real-time estimation is critical in various human-computer interaction applications, especially in virtual reality and augmented reality [20, 7, 23]. Recently, many studies achieved impressive progress by utilizing hand depth images from depth cameras. However, it still remains challenging to achieve accurate and real-time estimation, due to various issues such as self-occlusion, noise, high dimensionality, and various orientations of a hand [12, 9, 24, 6].

With the advancement of deep neural networks (DNNs), various DNN-based hand pose estimation techniques achieved powerful performances. In most of these techniques, 2D convolution neural networks (CNNs) have been adopted to perform direct hand depth image processing [40, 10, 14, 30, 3]. However, 2D CNNs cannot fully take advantage of 3D spatial information of the depth image, which is essential for achieving high accuracy. An intuitive solution is to discretize hand depth images into a 3D voxelized representation and perform 3D-to-3D inference using a 3D CNN [11, 24]. However, its critical limitation is the cubic growth of memory consumption with an increase in the image resolution [31]. Thus, application of 3D CNNs has been limited to low-resolution images, which may lead to lose of critical details for estimation.

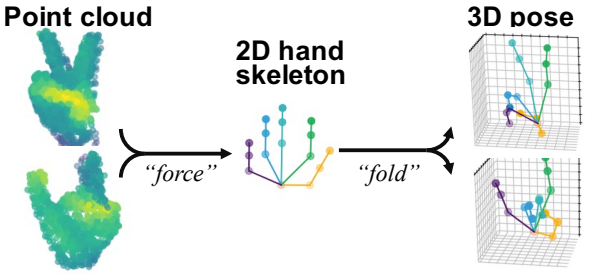

Figure 1. Illustration of the folding concept. The network can be interpreted as emulating the "force" through multi-scale features extracted from the point cloud. The "force" will drive a 2D hand skeleton to "fold" into the 3D joint coordinates representing the hand pose.

In contrast, the point cloud is being regarded as an efficient and precise representation for 3D hand pose estimation, as it models hand depth images into the continuous 3D coordinates without discretization. However, the point cloud could not be directly processed by conventional DNNs due to the irregular order of points, until the emergence of PointNet [28]. With a concise symmetric architecture composed of a point-wise shared-weights multi-layer perceptron (MLP) and a max-pooling layer, PointNet is invariant with the order of the input points.

Based on this architecture, a series of PointNet-based hand pose estimation models [9, 12, 4, 21] have been pro-

*Jong Hwan Ko is the corresponding author.

978-1-6654-2812-5/21/$31.00 ©2021 IEEE
DOI 10.1109/ICCV48922.2021.01107

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.

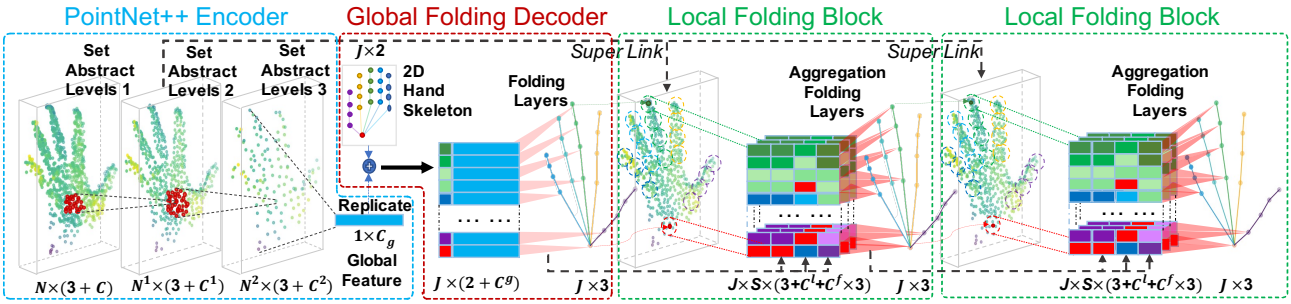

Figure 2. The HandFoldingNet architecture. It takes the preprocessed normalized point cloud with surface normal vectors from a 2D depth image as an input. The hierarchical PointNet encoder is then exploited to extract features of various levels to summarize a global feature from the input point cloud. The global folding decoder receives the global feature to guide the folding of a pre-defined 2D hand skeleton into the initial joint coordinates. In the end, the local features near the initial joint coordinates are grouped and fed into the local folding blocks to estimate the accurate joint coordinates.

posed. They can be summarized into two categories: 1) regression-based methods and 2) detection-based methods. Regression-based methods [9, 4] encode the hand shape into a single global feature through a PointNet-based feature extractor. The global feature representing the hand pose in the high dimensional latent space is fed into a non-linear regression network that performs inference of the joint coordinates. On the other hand, detection-based methods [12, 21] adopt hierarchical features to compute heat-map features for each point. The point-wise features represent the possibility distribution of each joint. However, the existing regression-based and detection-based strategies have limitations. The regression-based methods process only a single global feature, which is not sufficient for highly complex mapping into 3D hand poses. On the other hand, the detection-based methods propagate hierarchical features to each point including the points that contribute little to the specific joint estimation. Therefore, this redundant feature propagation significantly increases the computational cost and slows down the estimation.

To tackle these limitations, we propose HandFoldingNet, an accurate and efficient 3D hand pose estimation network. The key idea of HandFoldingNet is to fold a 2D hand skeleton into the 3D pose, guided by multi-scale features extracted from both global and local information. The motivation of adopting the folding-based design in FoldingNet [45] is that it is suitable for a 3D hand pose estimation task. Essentially, a specific hand pose is a result of applying a force on the human hand skeleton. The folding operation can be interpreted as emulating the "force" applied to the fixed 2D hand skeleton, as shown in Figure 1. In order to guide folding, HandFoldingNet introduces two novel modules that handle different scales of features: 1) a global-feature guided folding (**global folding**) decoder and 2) a joint-wise local-feature guided folding (**local folding**) block. Inspired by FoldingNet, a global folding decoder folds a 2D hand skeleton into the 3D hand joint coordinates. The global feature that guides folding is extracted from the input hand point cloud by a PointNet-based encoder [29, 9, 12]. The local folding block utilizes local features as well as spatial dependencies between the joints, in order to augment joint-wise features and correct the coordinate estimation. Utilization of local features is supposed to compensate for the weakness of conventional regression-based methods. Additionally, unlike the detection-based methods that propagate local features to all the points, we only extract a small region of local features near each joint, in order to avoid massive computations.

We evaluate our network on ICVL [36], MSRA [35] and NYU [40] datasets, which are challenging benchmarks commonly used for evaluation of a 3D hand pose estimation task. The results show that our network generally outperforms the previous state-of-the-art methods in terms of both accuracy and efficiency. The proposed network achieves the mean distance errors of 5.95mm, 7.34mm and 8.58mm on the ICVL, MSRA and NYU datasets, respectively. Meanwhile, it contains only 1.28M parameters and runs in real-time with 84 frames per second on a single GPU.

The key contributions of this paper are as follows:

- We propose a novel neural network, HandFoldingNet, which takes the hand point cloud as input and estimates the 3D hand joint coordinates based on the multiscale-feature guided folding.
- We propose a global-feature guided folding decoder that infers joint-wise features and coordinates. The joint-wise features help the model exploit natural spatial dependencies between the joints for better estimation performance.
- We propose joint-wise local-feature guided folding to capture local features and spatial dependencies that augments joint-wise features for higher accuracy.
- We conduct extensive experiments to analyse the efficiency and accuracy of our proposed network and its key components.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.

2. Related Work

2.1. Depth-based 3D Hand Pose Estimation

Traditional 3D hand pose estimation approaches based on depth images are mainly implemented in three categories: generative methods [18, 41, 39, 32], discriminative methods [17, 22], and hybrid methods [38, 34, 37]. In recent years, DNN-based models showed superior performance on 3D hand pose estimation tasks. Representative 2D CNNs are commonly adopted to pose estimation in various implementations. A series of studies [40, 10] exploited 2D CNNs in order to extract a 2D heat-map that represents the possibility distribution of hand joints from a depth image.

Another line of work proposed regression-based methods based on 2D CNNs [14, 30, 3], which act as feature extractors that provide efficient features for joint coordinates regression. Instead of processing in the 2D space, several approaches [11, 24] encoded 2D depth images into 3D voxels and adopted 3D CNNs to estimate the 3D hand pose. As depth images can be easily transformed into the point cloud by multiplying the camera intrinsic matrix, several point cloud based models [9, 12, 4, 21] have been proposed. They showed acceptable efficiency and performance by directly processing the input coordinates to estimate the joint coordinates in the identical 3D space.

HandFoldingNet is inspired by these point cloud based methods, but it differs from them in the following aspects. The proposed network does not directly regress the hand joint coordinates nor estimate the point-wise probability distribution. Instead, it first regresses the initial joint coordinates for grouping local features. Meanwhile, it also provides joint-wise features for modeling spatial dependencies. In the end, the network aggregates these local features and spatial dependencies to estimate the accurate joint coordinates.

2.2. Deep Point Cloud Reconstruction

Deep point cloud reconstruction aims to reconstruct the point cloud based on the features extracted from images, point clouds, or other types of data. An intuitive way of achieving the point cloud reconstruction is to adopt 3D CNNs, as in [44, 2, 13, 33]. However, these approaches reconstruct the voxelized representation of the point cloud. Instead of CNN-based methods, other approaches [1, 45, 43, 5] proposed direct reconstruction of the point cloud.

Theoretically, our main task, estimating hand joint coordinates for a given hand point cloud, can be transformed into the point cloud reconstruction task, because the estimated joint coordinates can be treated as a small set of points that need to be reconstructed. Therefore, we inherit the idea of FoldingNet [45] to reconstruct the joint point cloud. FoldingNet proposed a novel folding operation implemented by a sequence of shared-weights MLPs. This folding operation can be intuitively interpreted as learning the "force" to fold a given 2D grid lattice into the target point cloud. There are two critical differences between our network and FoldingNet: 1) we introduce folding of a 2D hand skeleton instead of a regular grid lattice in order to adapt it to the hand pose estimation task, 2) we exploit multi-scale features for higher estimation accuracy, unlike FoldingNet that processes only a single global feature.

3. HandFoldingNet

HandFoldingNet aims to perform hand pose estimation using 2D hand joint skeleton folding. The network architecture is shown in Figure 2. It takes an $N \times 6$ matrix $(\mathbf{P}^{nor}, \mathbf{F}^{nor})$, which represents a set of normalized points, as an input. Each row of the input matrix is composed of a normalized 3D xyz coordinate $\mathbf{p}_i^{nor} \in \mathbf{P}^{nor}$ and the corresponding 3D surface normal vector $\mathbf{f}_i^{nor} \in \mathbf{F}^{nor}$. The output is a $J \times 3$ matrix, representing the 3D coordinates of estimated J joints. The N points are firstly input to the hierarchical PointNet encoder that extracts local features of various levels and a single global feature. Then the global feature is fed into the global-feature guided folding decoder and guides folding of the fixed 2D hand skeleton into the 3D joint coordinates. In order to augment the estimation performance, the output from the global folding decoder and local features near them are processed by joint-wise local-feature based folding blocks.

3.1. Point Cloud Preprocessing

First, the 2D depth image is converted into a point cloud by reprojecting the pixels in the 3D space, forming the model input $(\mathbf{P}^{nor}, \mathbf{F}^{nor})$. We follow the point cloud preprocessing method described in HandPointNet [9]. The input depth images are first transformed into point cloud representations through camera intrinsic parameters, to adapt to our point cloud based network. Then, in order to deal with various hand orientations, an oriented bounding box (OBB) is created from the 3D point cloud. After that, the point cloud is rotated into the OBB coordinate system, whose axes are aligned with the principle components of the hand points distribution. The oriented points are subsampled and normalized into the range of [-0.5, 0.5] to form the final input coordinates $\mathbf{P}^{nor}$. In the end, point-wise surface normal vectors $\mathbf{F}^{nor}$ are calculated from the normalized point cloud. Please refer [9] for more details.

3.2. Hierarchical PointNet Encoder

We exploit the same hierarchical PointNet encoder as in [9, 12] to extract features from the unordered point cloud. As shown in Figure 2, the encoder consists of a cascade of L point set abstraction levels. The l-th level $(l \in \{1, 2, \ldots, L\})$ takes $N^{l-1} \times (3 + C^{l-1})$ matrix from the

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.

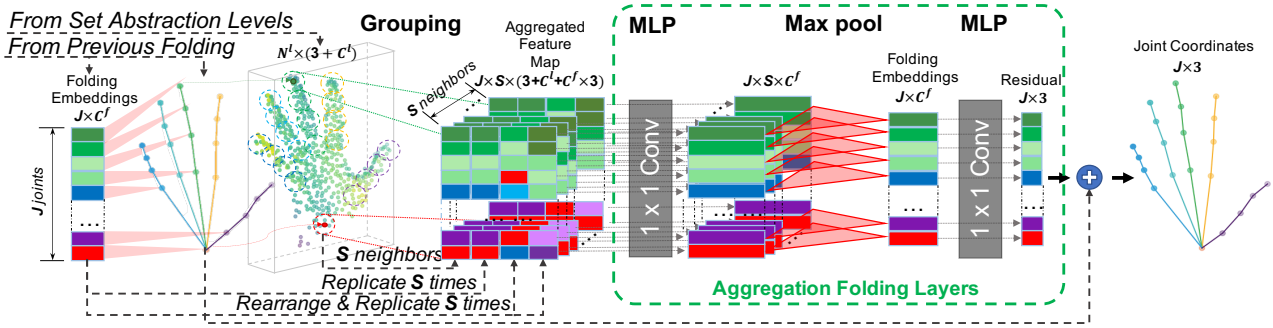

Figure 3. Joint-wise local feature guided folding block. The local folding block accepts three inputs, which are the previously estimated joint coordinates, folding embeddings from intermediate layers of the previous folding block, and a local feature map extracted by the previous set abstraction level. The joint coordinates are used as centroids that group local features from the local feature map. Folding embeddings are rearranged to be aligned with the corresponding adjacent joints to collect spatial dependencies. Ultimately, the aggregated feature map composed with grouped local features and rearranged embeddings is fed into a symmetric architecture to compute the residual with respect to the previously-estimated joint locations for more accurate joint estimation.

previous $(l-1)$-th level as an input, of which the i-th row is composed of a 3D coordinate $\mathbf{p}_i^{l-1}$ and the corresponding feature $\mathbf{f}_i^{l-1}$. Then it outputs $N^l \times (3+C^l)$ matrix, which is composed of N^l of sub-sampled centroids $\mathbf{p}_i^l$ and their corresponding C^l-dim local features $\mathbf{f}_i^l$. Specifically, for the first level, the input coordinate is $\mathbf{p}_i^{nor}$ and the corresponding feature is a 3D surface normal vector $\mathbf{f}_i^{nor}$.

The N^l centroids are randomly sampled from the input coordinates. Then, S neighbor points with their corresponding features around each centroid $\mathbf{p}_i^l$ are gathered as a local region $\{\mathbf{p}_{s,i}^{l-1}, \mathbf{f}_{s,i}^{l-1}\}_{s=1}^{S}$ by using the ball query [29] within a specified radius r. The coordinates in the local region are then translated to the local frame relative to their centroid: $\mathbf{p}_{s,i}^{l-1} - \mathbf{p}_i^l$. For each local region, a symmetric PointNet [28] with a 3-layer MLP is adopted to generate a C^l-dim feature for each point in the region. Subsequently, a max-pooling operation aggregates these point-wise features into a single local feature representing the corresponding centroid. Therefore, the local feature of the j-th sub-sampled centroid in the l-th level is represented as:

$$\mathbf{f}_i^l = \underset{1 \le s \le S}{MAX}(h([\mathbf{p}_{s,i}^{l-1} - \mathbf{p}_i^l, \mathbf{f}_{s,i}^{l-1}])), \tag{1}$$

where h is the MLP, MAX is the channel-wise max-pooling operation, and '$[\cdot]$' is the concatenation operation.

For the last level, it directly adopts the shared-weights MLP and max-pooling operation on the whole input (without sampling) in order to generate the single C^g-dim global feature, which is represented as:

$$\mathbf{g} = \underset{1 \le i \le N^{L-1}}{MAX}(h([\mathbf{p}_i^{L-1}, \mathbf{f}_i^{L-1}])). \tag{2}$$

3.3. Global-Feature Guided Folding Decoder

The proposed decoder folds a fixed 2D hand skeleton into the 3D coordinates of joints, being guided by a global feature. The hand skeleton is a set of hand joint coordinates in a 2D plane and is handcrafted by the following steps: 1) randomly choosing samples from the training set, 2) measuring the average length of links between each pair of adjacent ground truth joints from the samples, 3) unfolding links in a 2D plane, 4) collecting the coordinates of joints across every two connected links. An example of the 2D hand skeleton for the ICVL dataset is shown in Figure 4.

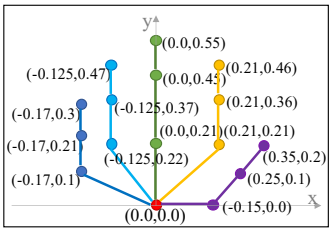

Figure 4. An example of a 2D hand skeleton based on the ICVL dataset. The skeleton contains $J = 16$ points, each of which is represented as a 2D coordinate.

After the hierarchical PointNet encoder extracts the global feature $\mathbf{g}$, it is fed into the global folding decoder. Before inserting the global feature $\mathbf{g}$, we replicate it J times and concatenate the replicated features with the fixed hand skeleton, whose size is $J \times 2$. The result of the concatenation is supplied to a 2-layer MLP that generates a high-dimensional folding embedding $\mathbf{e_j}$ for each joint. A subsequent 1-layer MLP predicts the initial 3D joint coordinates by processing input embeddings. Hence, the output coordinate $\mathbf{j}_j^0$ of the j-th joint is represented as:

$$\mathbf{j}_j^0 = h_p(\mathbf{e}_j) \text{ where } \mathbf{e}_j = h_e([\mathbf{skel}_j, \mathbf{g}]), \tag{3}$$

where h_p and h_e denote the MLPs, $\mathbf{e}_j$ denotes the intermediate folding embedding, and $\mathbf{skel}_j$ denotes the j-th point of 2D coordinate of the fixed skeleton.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.

3.4. Joint-Wise Local-Feature Guided Folding Block

Using only a single global feature (*i.e.* global-feature guided folding and other regression-based methods) is not sufficient to accurately estimate the joint coordinates . We believe that the use of additional joint-wise local features encourages the network to correct the joint coordinates.Therefore, we propose a novel joint-wise local-feature guided folding block for capturing local features and spatial dependencies that help better estimation.

As shown in Figure 3, the output coordinates from the $(k-1)$-th folding block are firstly used as centroids for the current k-th local folding block. The J centroids group J local regions from the output of the l-th set abstraction level within radius r. From each region, S neighbors are sampled, each of which is composed of a 3D local coordinate $\mathbf{p}^l_{s,j} - \mathbf{j}^{k-1}_j$ and a C^l-dim corresponding local feature $\mathbf{f}^l_{s,j}$, where $1 \leq s \leq S$. Therefore, the output size of this grouping is $J \times S \times (3 + C^l)$. Note that l is set to 1 as default, while the selection of l will be discussed in Section 4.4.

In addition, we introduce a rearrangement process that explicitly models spatial dependencies. It is worth mentioning that, the feature of a specific joint is represented by the corresponding row of the folding embeddings from the global folding decoder. Similarly, the local folding block provides joint-wise folding embeddings as well, enabling the network to stack more local folding blocks for accurate estimation. The rearrangement process first permutes the folding embeddings in order to form rearranged embeddings, which match the spatial dependency mapping as shown in Figure 5. The j-th row of each rearranged embedding is the folding embedding of the adjacent joints of the j-th joint. Then, we form the spatial dependency feature map by concatenating rearranged embeddings with the input folding embeddings. In the dependency mapping, as shown in Figure 5, each joint links with the other two adjacent joints. Therefore, this rearrangement process takes the folding embeddings of size $J \times C^f$ and outputs a spatial dependency map with size $J \times (C^f + C^f + C^f)$. Specifically, since the fingertips only have one adjacent joint, we concatenate them with themselves to keep a uniform shape of the spatial dependency map. As shown in Figure 5, there are self-relations for fingertips. Moreover, we replicate the spatial dependency feature map S times to align the dimension with the previous grouping output before the following aggregation.

After local features and the spatial dependency feature map are prepared, we concatenate them together, to form an aggregated feature map. The aggregated feature map is then fed to aggregation folding layers with symmetric structure, as shown in Figure 3. In this structure, we introduce a 3-layer MLP and a max-pooling, which aggregate the features into a single folding embedding for each joint.

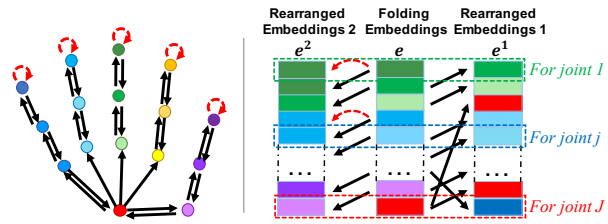

Figure 5. The spatial dependency mapping between hand joints of the ICVL dataset (left). Each joint permutes its embedding $\mathbf{e}_j$ to map with its two adjacent joints along the mapping direction of the arrows forming two rearranged embeddings $\mathbf{e}^1_j$ and $\mathbf{e}^2_j$ (right). Exceptionally, fingertips are forced to map with themselves (red dotted arrows) to keep consistency.

Block type	r	S	N^l	MLP channels	max
SA (l=1)	0.12	64	512	32, 32, 128	√
SA (l=2)	0.2	64	128	64, 64, 256	√
SA (l=3)	-	128	1	128, 128, 512	√
global fold (k=0)	-	-	J	256, 256, 3	×
local fold (k=1)	0.4	64	J	256, 256, 256	√
	-	-	J	256, 256, 3	×
local fold (k=2)	0.4	64	J	256, 256, 256	√
	-	-	J	256, 256, 3	×

Table 1. Implementation specifications. Each block contains four types of hyperparameters: search radius (r), the number of grouping neighbors (S), sampling centroids (N^l), and the number of output channels of each MLP layer. Max stands for the existence of a max-pooling layer at the end of the block. SA stands for the set abstraction level of PointNet encoder. The local folding blocks are divided into two parts at max-pooling for the clear representation.

Subsequently, we introduce another 3-layer MLP that maps the high-dimensional embedding into the 3D coordinates. Intuitively, since each joint focuses on its individual local region, only a relative displacement can be effectively computed by this MLP-MAX-MLP structure. Therefore, we inherit the residual block design [15]. The final joint coordinates are calculated by adding relative displacement outputs with the previously predicted coordinates. Hence, the j-th estimated joint of the k-th block is represented as:

$$\mathbf{j}^k_j = h_r(\underset{1 \leq s \leq S}{MAX}(h_f([\mathbf{p}^l_{s,j} - \mathbf{j}^{k-1}_j, \mathbf{f}^l_{s,j}, \mathbf{e}^{k-1}_j]))) + \mathbf{j}^{k-1}_j, \quad (4)$$

where h_r and h_f denote the shared-weights MLPs. $\mathbf{j}^{k-1}_j$ indicates the j-th output joint coordinate of the previous global folding decoder or local folding block. $\mathbf{p}^l_{s,j}$ and $\mathbf{f}^l_{s,j}$ are the s-th neighbor coordinate and feature of the j-th joint where l denotes the l-th set abstraction level. $\mathbf{e}^{k-1}_j$ indicates the concatenation of the j-th row of the folding embeddings and its two adjacent joints embeddings from the previous global folding decoder or the local folding block.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.

3.5. Loss Function

As our loss function, we adopt smooth L1 loss, which is less sensitive to outliers than L2 loss. The smooth L1 loss is defined as

$$L1_{smooth}(\mathbf{x}) = \begin{cases} 0.5|\mathbf{x}|, & |\mathbf{x}| < 0.01 \\ |\mathbf{x}| - 0.005, & otherwise \end{cases}. \quad (5)$$

Since the global folding and local folding blocks of our network output their respective estimated coordinates, we supervise all outputs by the following joint loss function:

$$\mathcal{L} = \sum_{j=1}^{J} L1_{smooth}(\mathbf{j}_j^0 - \mathbf{j}_j^*) + \sum_{k=1}^{K}\sum_{j=1}^{J} L1_{smooth}(\mathbf{j}_j^k - \mathbf{j}^*), \quad (6)$$

where $\mathbf{j}_j^*$ indicates the ground-truth coordinate of the j-th joint, and K indicates the quantity of stacked local folding blocks.

4. Experiments

4.1. Experiment Settings

We conducted experiments on an NVIDIA TITAN RTX GPU with PyTorch. For training, we used the Adam optimizer [19] with beta1 = 0.5, beta2 = 0.999, and learning rate α = 0.001. The number of input points to the network was preprocessed to 1,024 and the batch size was set to 32. The network implementation details are shown in Table 1. Batch normalization [16] and the ReLU [25] activation function are adopted in all MLP layers except the layers that output coordinates and residuals. Meanwhile, to avoid overfitting, we adopted online data augmentation with random rotation ([-37.5, 37.5] degrees around z-axis), 3D scaling ([0.9, 1.1]), and 3D translation ([-10, 10]mm). We evaluated the performance of the proposed model using public hand pose datasets, the ICVL [36], MSRA [35] and NYU [40] datasets. We trained the model for 400 epochs on ICVL, 200 epochs on NYU and 80 epochs (with a learning rate decay of 0.1 after 60 epochs) on MSRA.

4.2. Datasets and Evaluation Metrics

MSRA Dataset. The MSRA dataset [35] provides more than 76K frames from 9 subjects. Each subject contains 17 hand gestures. The ground truth of each frame contains $J = 21$ joints, including one joint for a wrist and four joints for each finger. Following the most recent work [35], we evaluate this dataset with the leave-one-subject-out cross-validation strategy.

ICVL Dataset. The ICVL dataset [36] is a commonly-used depth stream hand pose dataset that provides 22K and 1.6K depth frames for training and testing, respectively.

Methods	Mean error (mm)			Input	Type
	ICVL	MSRA	NYU		
DeepModel [46]	11.56	-	17.04	2D	R
DeepPrior [27]	10.4	-	19.73	2D	R
Ren-4x6x6 [14]	7.63	-	13.39	2D	R
Ren-9x6x6 [42]	7.31	9.7	12.69	2D	R
DeepPrior++ [26]	8.1	9.5	12.24	2D	R
Pose-Ren [3]	6.79	8.65	11.81	2D	R
DenseReg [42]	7.3	**7.2**	10.2	2D	D
CrossInfoNet [6]	6.73	7.86	10.08	2D	R
JGR-P2O [8]	6.02	7.55	**8.29**	2D	D
3DCNN [11]	-	9.6	14.1	3D	R
SHPR-Net [4]	7.22	7.76	10.78	3D	R
HandPointNet [9]	6.94	8.5	10.54	3D	R
Point-to-Point [12]	6.3	7.7	9.10	3D	D
V2V [24]	6.28	7.59	8.42	3D	D
Ours	**5.95**	**7.34**	8.58	3D	R

Table 2. Comparison of the proposed method with previous state-of-the-art methods on the ICVL, MSRA and NYU datasets. Mean error indicates the mean distance error. Input indicates the input representation of 2D (depth image) or 3D (voxel or point cloud). Type D and R indicate the detection-based method and regression-based method, respectively.

The ground truth of each frame contains $J = 16$ joints, including one joint for a palm and three joints for each finger. Since the frames also contain the human body area, we firstly crop the hand area from a depth image with the method proposed in [26], and take the output joint locations of the global folding decoder to segment the image of the hand area.

NYU Dataset. The NYU dataset is captured from three different views. Each view contains 72K training 8K testing depth images captured with the Microsoft Kinect sensor. Following recent works, we only use one view and 14 joints out of total of 36 annotated joints for training and testing. We also follow the same hand area segmenting process as in the ICVL dataset.

Evaluation metrics. We evaluate the hand pose estimation performance with two commonly-used metrics: the mean distance error and the success rate. The mean distance error measures the average Euclidean distance between the estimated coordinates and ground-truth ones for all the joints over the entire testing set. The success rate is the fraction of the frames whose mean distance error is less than a certain distance threshold.

4.3. Comparison with State-of-the-arts

We compare HandFoldingNet with other state-of-the-art methods, including methods with 2D (depth image) input: model-based method (DeepModel) [46], DeepPrior [27], improved DeepPrior (DeepPrior++) [26], region ensemble network (Ren-4x6x6 [14], Ren-9x6x6 [42]), Pose-Ren [3], dense regression network (DenseReg) [42], CrossInfoNet

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.

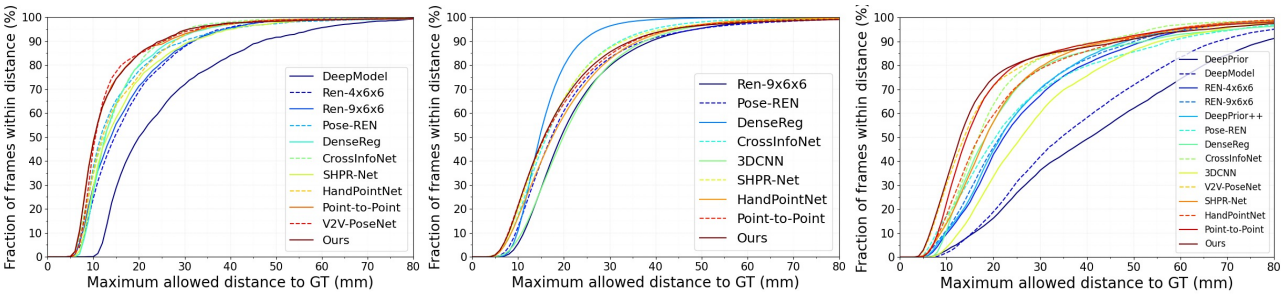

Figure 6. Comparison with the state-of-the-art methods using the ICVL (left), MSRA (middle) and NYU (right) dataset. The success rate is shown in this figure.

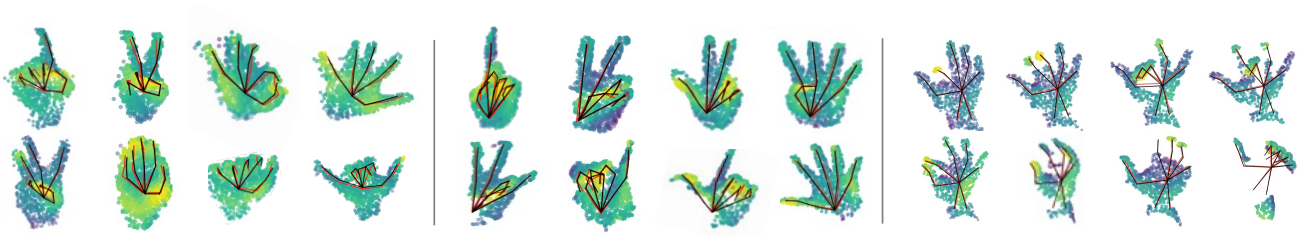

Figure 7. Qualitative results of HandFoldingNet on the ICVL (left), MSRA (middle) and NYU (right) dataset. Hand depth images are transformed into 3D points as shown in the figure. Ground truth is shown in black, and the estimated joint coordinates are shown in red.

[6] and JGR-P2O [8], and methods with 3D (point cloud or voxel) input: 3DCNN [11], SHPR-Net [4], HandPointNet [9], Point-to-Point [12] and V2V [24]. Figure 6 shows the success rate on the ICVL, NYU, and MSRA dataset. The qualitative results are represented in Figure 7.

Table 2 summarizes the performance based on the mean distance error on the three datasets. The results show that our method outperforms the existing methods on the ICVL dataset, achieving the mean distance error of 5.95mm. The proposed model also achieves the second-lowest error on the MSRA dataset and third-lowest error on the NYU dataset. Among methods using the 3D input, our method outperforms other state-of-the-art methods on both ICVL and MSRA datasets. Also, HandFoldingNet shows the state-of-the-art performance among regression-based methods on all three datasets. Figure 6 represents that our method achieves the highest success rate when the error threshold is lower than 10mm, 13mm and 25mm on the ICVL, MSRA and NYU datasets, respectively.

4.4. Ablation Study

We conduct ablation experiments evaluating the performance impact of each component in our model. The following experiments are evaluated based on the ICVL dataset.

Effectiveness of the local folding block. This experiment evaluates the accuracy improvement by attaching the proposed local folding block. To compare with the proposed network having one global folding and two local folding blocks (triple fold), we introduce a shallow network (single fold) that only provides the global folding, a network with only one local folding block (double fold), and a network with three local folding blocks (quadra fold). Table 3 shows the performance comparison between the models with different number of local folding.

The result shows that local folding significantly reduces the distance error. This experiment proves that the global folding that only accepts a single global feature for estimation is relatively weak, and the local features contributes the correction of the final joint coordinates. Although attaching more local folding blocks increases the inference overhead, the number of parameters and operations of the proposed model (triple fold) are not significant compared to the existing models, as analyzed in Section 4.5. However, the result also shows that the model performance is saturated at triple fold. The reason is that the additional gradients from the third local fold corrupt the back propagation and make the training harder. Note that double fold still outperforms several point cloud based networks with smaller parameter size and operation count.

Effectiveness of local features and spatial dependencies. We evaluate the contribution of the critical feature components of the aggregated feature map, which are the local feature and spatial dependency feature. We conduct two independent experiments: 1) *without local feature* and 2) *without spatial dependency*. For *without local feature*, we remove the grouped local feature component of the aggregated map and maintain the spatial dependency component. For *Without spatial dependency*, we remove rearranged folding embeddings and maintain the local feature. Table 4 shows that the mean distance error increases by

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.

Global fold	# Local fold	Mean error (mm)	#Params	FLOPs
√	×	8.13	0.38M	0.46G
√	1	6.34	0.78M	0.78G
√	2	**5.95**	1.28M	1.10G
√	3	6.08	1.78M	1.48G

Table 3. Comparison of different numbers of local folding blocks used in the model. # Local fold indicates the number of local folding blocks attached after the global folding decoder. # Params indicates the total number of parameters of the network. FLOPs indicates the total number of floating-point operations required for the network inference.

Local feature	Spatial dependency	Mean error (mm)	# Params	FLOPs
×	√	7.90	1.21M	1.04G
√	×	6.35	1.08M	0.91G
√	√	**5.95**	1.28M	1.10G

Table 4. Comparison of different settings between the local feature and spatial dependency.

Sampling level	Mean error (mm)	#Params	FLOPs
input	6.58	1.21M	1.04G
first (l=1)	**5.95**	1.28M	1.10G
second (l=2)	6.48	1.34M	1.17G

Table 5. Comparison of different set abstraction levels for local features.

Methods	# Param	Speed	Time (ms)	GPU Type
V2V-PoseNet [24]	457.5M	3.5	23 + 5.5	TITAN X
HandPointNet [9]	2.58M	48	8.2 + 11.3	GTX1080
Point-to-Point [12]	4.3M	41.8	8.2 + 15.7	TITAN XP
Ours	**1.28M**	**84**	**8.2 + 3.7**	TITAN RTX

Table 6. Comparison of the model size and inference time for the methods using the 3D input. Speed stands for the frame rate (fps) on a single GPU. Time stands for the total computation time including preprocessing time and model inference time.

1.55mm without the local features. Similarly, without the spatial dependency, the mean distance error increases by 0.40mm. These experiments show that the both features are critical for improving estimation accuracy. Meanwhile, the local feature contributes to the performance more efficiently, as it requires smaller parameters and FLOPs while achieving better performance than using the spatial dependency.

Sampling level of local features. HandFoldingNet is composed of three set abstraction levels in the PointNet encoder, where each level has different input points density and feature complexity. Therefore, we should carefully determine the abstraction level so that the local folding blocks can effectively collect extra local features. To analyze the performance impact of the abstraction level, we experiment with the input, first, and second set abstraction levels as the input to the local folding blocks. Table 5 indicates that adopting the output point cloud from the first set abstraction level achieves the highest performance because the neighbor points around the joints are adequate (input points are dense) and the features they provide are effectively informed (input features are complex). On the other hand, the input point cloud is not complex enough as it only includes 3D surface normal vectors. Consequently, directly using the input point cloud for local folding is not effective in capturing necessary features that can improve the performance. Conversely, using higher abstraction level (sampling level 2) degrades the performance. Although the second level features are sufficiently complex, the points are actually sparse in the 3D space. Therefore, the local folding can not group enough points.

4.5. Runtime and Model Size

The runtime of HandFoldingNet measured on an NVIDIA TITAN RTX GPU is 11.9ms per point frame in average, including 8.2ms for preprocessing and 3.7ms for network inference. Thus, it can run in real-time at about 84.0fps. Table 6 shows our method has the lowest total latency among the 3D-input based methods. Our method also achieves the fastest inference within the point cloud based methods that require 8.2ms of preprocessing time. Moreover, the number of parameters of our proposed network is sufficiently small, which is only 1.28M. Compared with previous state-of-the-art models, our model requires the least parameters.

5. Conclusion

In this paper, we proposed HandFoldingNet, a novel and efficient neural network that takes the point cloud as the input and estimates the 3D hand pose. The proposed network achieves the accurate joint coordinates estimation by leveraging the multi-scale features, including the global feature and the joint-wise local feature. Experimental results on three challenging benchmarks showed that our network outperforms previous state-of-the-art methods while requiring the minimal computational resources. Ablation experiments demonstrated the contribution of its key components for better accuracy and efficiency.

Acknowledgement

This work was partly supported by the Institute of Information and Communication Technology Planning & Evaluation (IITP) grant on AI Graduate School Program (IITP-2019-0-00421) and ICT Creative Consilience program (IITP-2020-0-00821) funded by the Korea government. Wencan Cheng was supported by the China Scholarship Council (CSC).

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49, 2018. 3

[2] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016. 3

[3] Xinghao Chen, Guijin Wang, Hengkai Guo, and Cairong Zhang. Pose guided structured region ensemble network for cascaded hand pose estimation. *Neurocomputing*, 395:138–149, 2020. 1, 3, 6

[4] Xinghao Chen, Guijin Wang, Cairong Zhang, Tae-Kyun Kim, and Xiangyang Ji. Shpr-net: Deep semantic hand pose regression from point clouds. *IEEE Access*, 6:43425–43439, 2018. 1, 2, 3, 6, 7

[5] Wencan Cheng and Sukhan Lee. Point auto-encoder and its application to 2d-3d transformation. In *International Symposium on Visual Computing*, pages 66–78. Springer, 2019. 3

[6] Kuo Du, Xiangbo Lin, Yi Sun, and Xiaohong Ma. Cross-infonet: Multi-task information sharing based hand pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9896–9905, 2019. 1, 6, 7

[7] Ali Erol, George Bebis, Mircea Nicolescu, Richard D Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, 2007. 1

[8] Linpu Fang, Xingyan Liu, Li Liu, Hang Xu, and Wenxiong Kang. Jgr-p2o: Joint graph reasoning based pixel-to-offset prediction network for 3d hand pose estimation from a single depth image. In *European Conference on Computer Vision*, pages 120–137. Springer, 2020. 6, 7

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

[10] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3593–3601, 2016. 1, 3

[11] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1991–2000, 2017. 1, 3, 6, 7

[12] Liuhao Ge, Zhou Ren, and Junsong Yuan. Point-to-point regression pointnet for 3d hand pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 475–491, 2018. 1, 2, 3, 6, 7, 8

[13] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016. 3

[14] Hengkai Guo, Guijin Wang, Xinghao Chen, Cairong Zhang, Fei Qiao, and Huazhong Yang. Region ensemble network: Improving convolutional network for hand pose estimation. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4512–4516. IEEE, 2017. 1, 3, 6

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 6

[17] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *European Conference on Computer Vision*, pages 852–863. Springer, 2012. 3

[18] Sameh Khamis, Jonathan Taylor, Jamie Shotton, Cem Keskin, Shahram Izadi, and Andrew Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2540–2548, 2015. 3

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[20] Rui Li, Zhenyu Liu, and Jianrong Tan. A survey on 3d hand pose estimation: Cameras, methods, and datasets. *Pattern Recognition*, 93:251–272, 2019. 1

[21] Shile Li and Dongheui Lee. Point-to-pose voting based hand pose estimation using residual permutation equivariant layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11927–11936, 2019. 1, 2, 3

[22] Hui Liang, Junsong Yuan, and Daniel Thalmann. Parsing the hand in depth images. *IEEE Transactions on Multimedia*, 16(5):1241–1253, 2014. 3

[23] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015. 1

[24] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE conference on computer vision and pattern Recognition*, pages 5079–5088, 2018. 1, 3, 6, 7, 8

[25] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 6

[26] Markus Oberweger and Vincent Lepetit. Deepprior++: Improving fast and accurate 3d hand pose estimation. In *Proceedings of the IEEE international conference on computer vision Workshops*, pages 585–594, 2017. 6

[27] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015. 6

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.

[28] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 4

[29] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 2, 4

[30] Pengfei Ren, Haifeng Sun, Qi Qi, Jingyu Wang, and Weiting Huang. Srn: Stacked regression network for real-time 3d hand pose estimation. In *BMVC*, page 112, 2019. 1, 3

[31] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017. 1

[32] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (ToG)*, 36(6):245, 2017. 3

[33] Abhishek Sharma, Oliver Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In *European Conference on Computer Vision*, pages 236–250. Springer, 2016. 3

[34] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3633–3642, 2015. 3

[35] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 824–832, 2015. 2, 6

[36] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3786–3793, 2014. 2, 6

[37] Danhang Tang, Jonathan Taylor, Pushmeet Kohli, Cem Keskin, Tae-Kyun Kim, and Jamie Shotton. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *Proceedings of the IEEE international conference on computer vision*, pages 3325–3333, 2015. 3

[38] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, et al. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016. 3

[39] Anastasia Tkach, Andrea Tagliasacchi, Edoardo Remelli, Mark Pauly, and Andrew Fitzgibbon. Online generative model personalization for hand tracking. *ACM Transactions on Graphics (ToG)*, 36(6):1–11, 2017. 3

[40] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):1–10, 2014. 1, 2, 3, 6

[41] Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision*, 118(2):172–193, 2016. 3

[42] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Dense 3d regression for hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2018. 6

[43] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3dn: 3d deformation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1038–1046, 2019. 3

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

[45] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 2, 3

[46] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. Model-based deep hand pose estimation. *arXiv preprint arXiv:1606.06854*, 2016. 6

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:23:00 UTC from IEEE Xplore. Restrictions apply.