HandR^{2}N^2: Iterative 3D Hand Pose Estimation Using a Residual Recurrent Neural Network

Wencan Cheng
Department of Artificial Intelligence
Sungkyunkwan University
cwc1260@skku.edu

Jong Hwan Ko
College of Information and Communication Engineering
Sungkyunkwan University
jhko@skku.edu

Abstract

3D hand pose estimation is a critical task in various human-computer interaction applications. Numerous deep learning based estimation models in this domain have been actively explored. However, the existing models follow a non-recurrent scheme and thus require complex architectures or redundant parameters in order to achieve acceptable model capacity. To tackle this limitation, this paper proposes HandR^{2}N^2, a compact neural network that iteratively regresses the hand pose using a novel residual recurrent unit. The recurrent design allows recursive exploitation of partial layers to gradually optimize previously estimated joint locations. In addition, we exploit graph reasoning to capture kinematic dependencies between joints for better performance. Experimental results show that the proposed model significantly outperforms the existing methods on three hand pose benchmark datasets in terms of both accuracy and efficiency. Codes and pre-trained models are publicly available at https://github.com/cwc1260/HandR2N2.

1. Introduction

3D hand pose estimation, which estimates the 3D positions of hand keypoints, provides a fundamental understanding of human hand activity. Thus, it is a critical task for many human-computer interaction applications such as robotics and augmented/virtual reality. With the assistance of deep learning techniques and low-cost depth cameras, numerous studies have contributed to the substantial progress in the field of hand pose estimation. However, accurate and efficient hand pose estimation is still challenging because of the various probability of hand orientations, severe self-occlusion, and noisy depth images [12, 9, 18, 6].

Most of the recent state-of-the-art approaches are based on deep learning, especially deep convolutional neural networks (CNNs). Such CNN-based approaches [31, 10, 13, 25, 1, 8, 23, 6, 23] generally accept 2D depth images as an input, which can be directly processed by the 2D convolutional layers. However, it is difficult for a 2D CNN to capture the highly non-linear mapping from a 2D depth image to the 3D hand pose, significantly restricting its efficiency and accuracy. Therefore, a line of works [11, 18] discretized 2D input images into 3D volumetric representations and subsequently applied a 3D CNN for direct 3D-to-3D inference. However, a critical drawback of a 3D CNN is its computational overhead that increases cubically with an increase of the input resolution [4]. In contrast, another series of hand pose estimation models [9, 12, 2, 15, 4] have presented remarkable performances by using PointNet [20, 21], which allows the models to accept a set of continuous 3D coordinates converted from an input hand image without discretization.

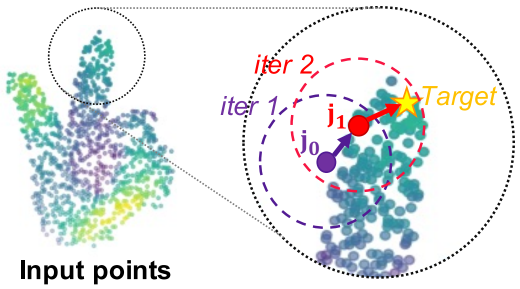

Figure 1. Illustration of the recurrent concept. For each iteration, our proposed recurrent architecture samples local points around joints from previous iteration to generate displacements to refine the joint locations.

Despite the accuracy improvement achieved by the aforementioned approaches, they still suffer from large and complex model architectures that require substantial computational overheads. Thus, these methods are not suitable for resource-limited devices such as wearable or hand-held devices. Furthermore, all the existing approaches follow a non-recurrent architecture. As a result, they are not flexible for varying resource constraints and accuracy targets.

In this work, we tackle these limitations using a recurrent architecture whose capacity can be dynamically adjusted to achieve either higher accuracy or higher efficiency by iteratively using a part of model parameters.

The recurrent structure can achieve the high capacity of deeper networks by the recursive use of fewer shared parameters [34]. Various image processing models that used a recurrent neural network (RNN) as an explicit iterative approach have already yielded encouraging results [30, 5, 35, 26]. To fully take advantage of the RNN in a hand pose estimation task, we propose a novel residual recurrent neural network, HandR^{2}N^2, that iteratively regresses accurate 3D hand pose from an input hand point cloud. Intuitively, HandR^{2}N^2 can be served as a general optimizer that recursively searches the optimal joint locations using the observation captured by the previous estimation, as shown in Figure 1.

To implement HandR^{2}N^2, we introduce two novel components: 1) a residual recurrent unit (RRU) and 2) an initialization module. The RRU is a key component that iteratively proceeds for updating its joint-wise hidden states and coordinate estimations. The initial joint-wise states and coordinates of the RRU are globally estimated by the initialization module.

The RRU specifically integrates with an attentive gate that can explore optimal points from a group of candidate points for the improved accuracy. Since the proposed RRU only computes the features contributed to the specific joints, it is more efficient than the similar prior model in [5], which re-computes the entire point features. Moreover, RRU exploits graph reasoning, which is based on a novel channel-wise graph convolutional network (GCN). GCN is capable to model the graph-structured data, making it particularly suitable for the kinematically structured hand data [8]. The use of the proposed channel-wise GCN enables the RRU to recursively enhance the hidden states by capturing the strong kinematic dependencies between joints. More importantly, the number of RRU iterations during inference can be dynamically scaled to be different from the training iteration. Specifically, the RRU inference iterations can be configured large to enhance the model capacity and accuracy, or small to reduce the computation requirement. Therefore, HandR^{2}N^2 is capable of adapting varying complexity and accuracy requirements as shown in Figure 2.

We evaluate HandR^{2}N^2 on three challenging benchmarks, ICVL [28], MSRA [27], and NYU [31] datasets. The results show that our network achieves a new state-of-the-art record. The proposed network reports the minimum mean distance errors of 5.70 mm, 6.42 mm and 7.27 mm on the ICVL, MSRA and NYU datasets, respectively. Meanwhile, it requires only 1.3 M parameters and runs with adjustable complexity 0.72+0.34$\times I$ GFLOPs, where I indicates the number of iterations. Thus, our model requires

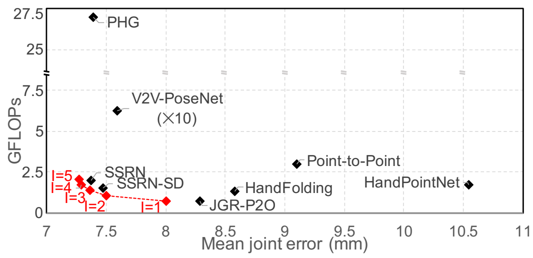

Figure 2. Comparison between operation count and estimation error. I is the iteration number of HandR^{2}N^2 during inference.

less computation for the highest accuracy compared with other state-of-the-art methods, as shown in Figure 2.

The key contributions of this paper are summarised as follows:

- We propose a novel iterative neural network architecture that takes the hand point cloud as input and progressively estimates the accurate 3D hand joint coordinates.
- We propose a novel residual recurrent unit that captures new local observations around previously generated coordinates to refine joint-wise hidden states and coordinate estimation. It also leverages a graph neural network to model dependency between the joints for performance improvement.
- We conduct extensive experiments to analyze the efficiency and effectiveness of our proposed network.

2. Related Work

2.1. Depth-based 3D Hand Pose Estimation

3D hand pose estimation approaches using depth images can be classified into three categories according to the input types: conventional 2DCNN-based methods that take raw 2D depth images as input [31, 10, 13, 25, 1, 8, 23, 6], 3DCNN-based methods that use 3D voxelized representations of depth images [11, 18], and PointNet-based methods that process the point cloud transformed from depth images [9, 12, 2, 15, 4]. In this section, we focus on PointNet-based methods, since the point cloud is an accurate and concise representation for efficient inference.

The point cloud can be easily acquired from depth images by multiplying the camera intrinsic matrix. However, due to its irregularity and disorder, processing point clouds has always been a challenging task until the emergence of PointNet [20, 21]. HandPointNet [9] was the first work that utilized PointNet as a backbone to directly regress the hand pose from the point cloud. Subsequently, the Point-to-Point model [12] used the hierarchical PointNet in order to generate the point-wise probability distribution for improved estimation. Based on these prior studies, SHPR-Net

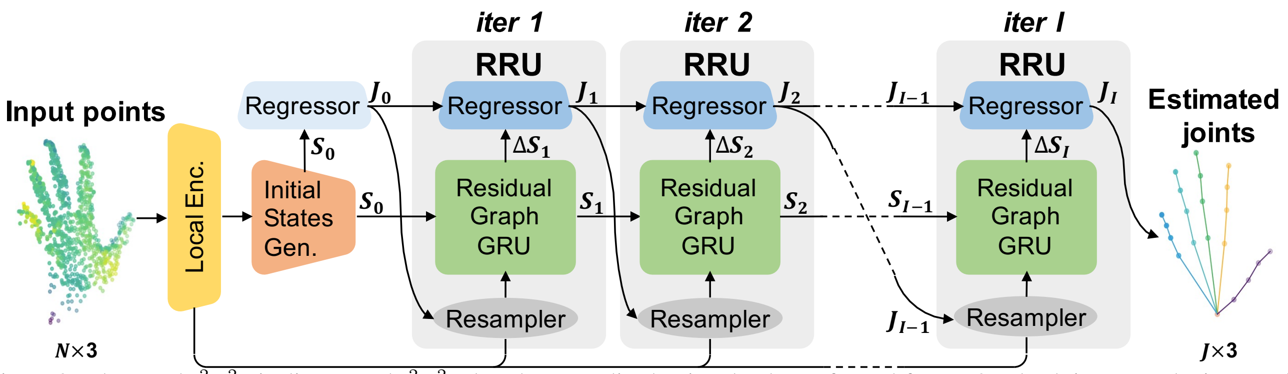

Figure 3. The HandR^{2}N^2 pipeline. HandR^{2}N^2 takes the normalized point cloud transformed from a 2D depth image as the input. The PointNet-based local encoder extracts local features from input points. The local features are fed into an initial states generator to initialize hidden states of the residual recurrent unit (RRU). The high-dimensional initialized hidden states are also transformed into 3-dimensional space to generate initial joint estimation. At last, the RRU iteratively resamples points around previously estimation joints to update hidden states and joint locations.

[2] combined HandPointNet with an auxiliary semantic segmentation sub-network to augment the performance. Point-to-Pose [15] further exploited residual permutation equivariant layers [22] to alleviate information loss caused by the max-pooling bottleneck. Recently, HandFoldingNet [4] introduced multiscale-feature-guided folding to reshape a predefined 2D hand skeleton into hand poses.

However, the existing methods mentioned above followed the single feed-forward design. In such feed-forward architecture, the model capacity can be increased by having a complex architecture or additional parameters. However, such solutions may result in overfitting and higher computational requirement. Unlike these methods, our model exploits the recurrent mechanism to iteratively refine accurate hand pose by reusing a part of parameters.

2.2. Recurrent Models for Hand Pose Estimation

Recently, recurrent models [7, 35, 33, 34, 36] have been successfully applied in 2D image based hand pose estimation. ACE-Net [7] and SeqHand-Net [35] exploited long short term memory (LSTM) to extract the temporal context information for more smooth and accurate estimations from color images. CADSTN [33] also applied LSTM to jointly model the spatiotemporal context from depth image sequences. However, as the input to these methods is a temporal image sequence, they cannot be used for hand pose estimation from a single image. In contrast, DIR-Net [34] iteratively reuses a part of its structure to refine a downsampled feature map of a single color image. Nonetheless, updating the whole feature map is computationally redundant, since the features without contribution, such as background, are also repeatedly computed. Thus, HCRNN [36] applies RNN on each individual joints for high inference efficiency.

Although the recurrent models have made significant progress in 2D image based hand pose estimation, they have not been actively studied in the point cloud based hand pose estimation. Taylor et. al. [29] proposed an online optimization method that iteratively optimizes a predefined hand template to fit hand points. A related research [5] iteratively repeated the entire alignment architecture to align hand pose from coarse global stages to fine local stages. It also employed LSTM among multiple global stages to refine global alignment. However, it is significantly computational heavy because it reuses the entire structure for multi-stage feature generation. Conversely, our network only captures a small part of local information around each joint for each iteration with partial parameters, making it more effective than re-computing global information. Furthermore, we integrate a graph reasoning module in the recurrent unit to enhance estimation by capturing kinematic dependencies between joints, which has not been adopted in [5].

3. Iterative 3D Hand Pose Estimation

HandR^{2}N^2 aims to perform iterative hand pose estimation based on a recurrent architecture, as shown in Figure 3. Intuitively, HandR^{2}N^2 refines the joint locations by iteratively searching the optimal position from the region around each joints, as illustrated in Figure 1. The input to HandR^{2}N^2 is a set of preprocessed 3D point coordinates $\mathbf{P} \in \mathbb{R}^{N\times 3}$ that represents the hand shape, which is transformed from a 2D depth image though the camera intrinsic matrix. The outputs are a sequence of the estimated 3D joints coordinates $\{\mathbf{J}_i \in \mathbb{R}^{J\times 3} | i = 0, 1, ..., I\}$, where $\mathbf{J}_I$ is our final estimation. The N points are firstly input to a local encoder that extracts local geometric features. Since the input point set is irregular and orderless, we exploit *furthest_point_sampling* method and a one-layer *set_conv_layer* [16] proposed by PointNet++ [21] to construct the local encoder $(\mathbb{R}^{N\times 3} \to \mathbb{R}^{(N/2)\times(3+d_{local})})$. Then, the local features are fed into the initial state generator to initialize the

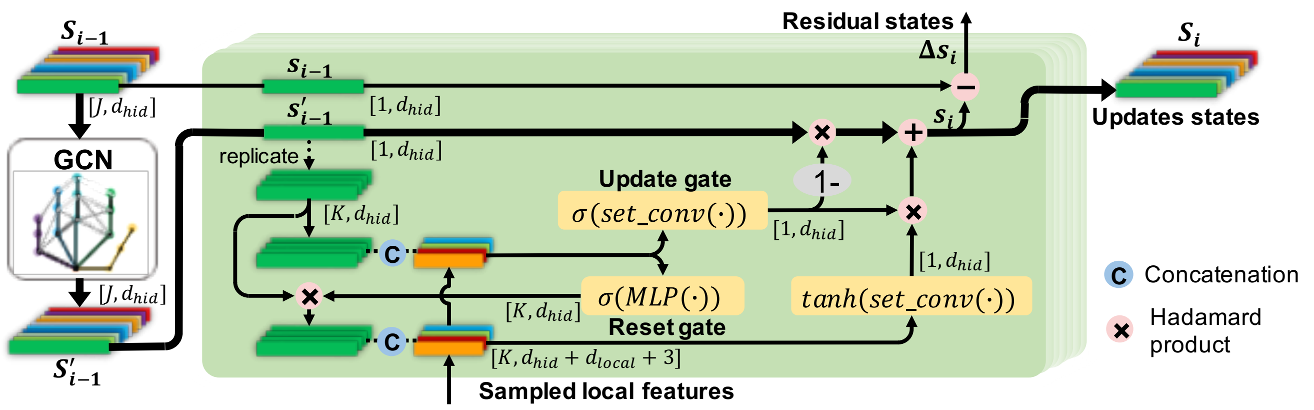

Figure 4. The architecture of the residual graph GRU. It first exploits a graph reasoning module to evolve hidden states. The evolved hidden states are subsequently updated by a GRU cell which takes resampled points as input. After updating, the residual between updated hidden states and previous states are output for joint refinement.

hidden states $\mathbf{S}_0 \in \mathbb{R}^{J \times d_{hid}}$, which are further processed by an initial joint regressor to obtain the initial estimation of joints $\mathbf{J}_0$. Afterward, a novel residual recurrent unit is unrolled I times to iteratively update the hidden states and joint estimations by resampling the useful local features around previously estimated joints. The number of iterations I is a hyper-parameter of the proposed network, and is quite flexible. Specifically, the number of training iterations (I_{tr}) can be different from the number of inference iterations (I_{in}), as discussed in Section 4.4.

3.1. Residual Recurrent Unit

HandR^{2}N^2 requires a recurrent unit capable of abstracting a set of input point features and capturing kinematic dependencies between joints, in order to perform iterative refinements that optimizes previous estimations. Obviously, the conventional recurrent units such as LSTM or GRU cannot satisfy these requirements, since their input and hidden state should be one single vector, which cannot accommodate the input point set. Therefore, we propose a novel component, a residual recurrent unit (RRU). The RRU consists of the following items: 1) a resampler, 2) a residual graph GRU cell, and 3) a residual regressor. The resampler samples local features around previously generated coordinates $\mathbf{J}_{i-1}$ forming the input point sets for the current i-th iteration. To update the hidden states with the input of point sets, we propose a novel residual graph GRU that implements its gates based on PointNet. In addition, the proposed GRU also cooperates with graph reasoning to augment the hidden states by capturing kinematic dependencies. At last, the residuals between the current and previous states are sent to the residual regressor to refine the previously generated coordinates.

Resampler. The resampler collects K-nearest-neighbor local points and their corresponding local features around each of previously estimated joint $\mathbf{j_{i-1}} \in \mathbf{J_{i-1}}$, as visualized in Figure 1. The neighbor points are then translated to a relative coordinate system with $\mathbf{j_{i-1}}$ as the origin.

Residual graph GRU. The architecture of the proposed residual graph GRU is visualized in Figure 4. Since many recent approaches [8, 23] suggested that a graph convolutional network (GCN) can model the beneficial dependencies among joints, the hidden states from the previous iteration are first augmented through a novel GCN, forming the evolved hidden states:

$$\mathbf{S'}_{i-1} = ReLU(\mathbf{A}\mathbf{S}_{i-1}\mathbf{W}), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{d_{hid} \times d_{hid}}$ is the trainable weights. Notably, $\mathbf{A} \in \mathbb{R}^{J \times J \times d_{hid}}$ is a novel channel-wise independent adjacent matrix, which means for each channel of S, $s_c \in \mathbb{R}^{\in J \times 1}$, there is an individual adjacency matrix $a_c \in \mathbb{R}^{J \times J}$ for the multiplication $AS = [a_1 s_1, a_2 s_2, ..., a_{d_{hid}} s_{d_{hid}}]$, where '$[\cdot,\cdot]$' is concatenation. Afterwards, the GCN-evolved hidden states are passed through a GRU cell to aggregate new information from the sampled local features and to update the hidden states for the current i-th iteration. Here, the conventional GRU cell is not suitable because the input features have one additional dimension than the hidden states. Therefore, we exploit *set_conv_layer* [16], which is composed of a shared multi-layer perceptron and a max-pooling layer, for gate computation to implement the GRU. First, the hidden states are replicated K times to be concatenated with input features. The concatenated output are then fed into each gate for computation. For those gates that directly modify values of the hidden states, we implement them with *set_conv_layer* to squeeze one dimension. Moreover, we implemented a novel attentive reset gate with a point-wise multi-layer perceptron (MLP) that enables the GRU to select good points for the hidden state update, which is beneficial for exploring the optimal joint

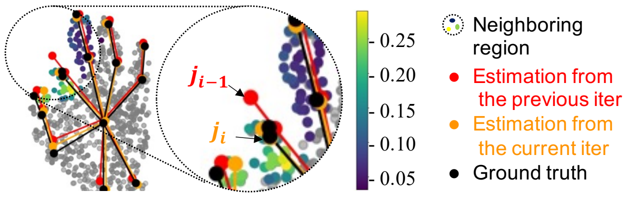

Figure 5. Visualization of the output from the reset gate in the proposed GRU. The small colored points in the circle are the grouped neighbors of the red input joint j_{i-1}, and their colors indicate the attentive intensity computed by Eq. 3. It shows that the points closer to the ground truth obtain relatively higher intensity. By leveraging clues to focus on highly-related points near the ground truth, the attentive GRU can output the accurate orange joint j_i.

locations, as shown in Figure 5. Thus, given the evolved hidden state $\mathbf{s'}_{i-1} \in \mathbf{S'}_{i-1}$ of a specific joint, and with the sampled local features $\{\mathbf{f}_k \in \mathbb{R}^{3+d_{local}} | 1 \leq k \leq K\}$, the GRU generates the updated hidden state $\mathbf{s}_i$:

$$\mathbf{z}_i = \sigma(\underset{1 \leq k \leq K}{set\_conv_z}([\mathbf{s'}_{i-1}, \mathbf{f}_k])), \tag{2}$$

$$\mathbf{r}_{i,k} = \sigma(MLP_r([\mathbf{s'}_{i-1}, \mathbf{f}_k])), \tag{3}$$

$$\tilde{\mathbf{s}}_i = tanh(\underset{1 \leq k \leq K}{set\_conv_h}([\mathbf{r}_{i,k} \odot \mathbf{s'}_{i-1}, \mathbf{f}_k])), \tag{4}$$

$$\mathbf{s}_i = (1 - \mathbf{z}_i) \odot \mathbf{s'}_{i-1} + \mathbf{z}_i \odot \tilde{\mathbf{s}}_i, \tag{5}$$

where σ is the sigmoid activation function, '$[\cdot, \cdot]$' is the concatenation operation, and $\odot$ is the Hadamard product. Note that, the updated hidden states encode the absolute joint locations in the latent space. However, we require relative representations to refine the previously generated joints. Therefore, we introduce a residual mechanism that outputs the relative representations of the updated information:

$$\mathbf{\Delta S}_i = \mathbf{S}_i - \mathbf{S}_{i-1}. \tag{6}$$

Residual regressor. The regressor accepts residual states of the current i-th iteration as input to refine the previous joints generated by the $(i-1)$-th iteration. The regressor is a linear transformation with a residual connection. Therefore, the joint regression of the current i-th iteration is represented as:

$$\mathbf{J}_i = \mathbf{\Delta S}_i \mathbf{W} + \mathbf{J}_{i-1}, \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{d_{hid} \times 3}$ is the trainable transformation matrix.

3.2. Initialization of the Hidden States and Joint Coordinates

The hidden states $\mathbf{S}$ of the RRU essentially are the embeddings that represent joints in the d_{hid}-dimensional latent space and can be further re-projected into the three-dimensional coordinates by the subsequent transformation. Thus, we inherit the idea of generating joint-wise embeddings to obtain the initialized hidden states $\mathbf{S}_0$, as proposed by HandFoldingNet [4]. The generation of the initial hidden states (joint-wise embeddings) are sequentially proceeded by two cascaded *set_conv_layers* and a three-layer bias-induced layer (BIL) [3]. Equivalent with HandFoldingNet [4], the cascaded *set_conv_layers* first down-sample and extract $N/8$ of local features and subsequently aggregate them as a global feature. The global feature is then replicated J times and fed into the BILs for the generation of hidden states for J joints. Finally, the J hidden states are injected into the regressor, which performs a linear transformation ($\mathbb{R}^{d_{hid}} \rightarrow \mathbb{R}^3$) for the initialization of the initial joint $\mathbf{J}_0$.

3.3. Training Loss Function

Following the previous works [25, 4], we adopt smooth L1 loss to supervise training because of its less sensitivity to the outliers. The smooth L1 loss is defined as:

$$L1_{smooth}(\mathbf{x}) = \begin{cases} 50\mathbf{x}^2, & |\mathbf{x}| < 0.01 \\ |\mathbf{x}| - 0.005, & otherwise \end{cases}. \tag{8}$$

Since the proposed network outputs an initial estimation and I iterative estimations, we supervise all of the outputs by the following joint loss function:

$$\mathcal{L} = \sum_{i=0}^{I} \sum_{j=0}^{J} \lambda^{I-i} L1_{smooth}(\mathbf{J}_{ij} - \mathbf{J}^*_j), \tag{9}$$

where $\mathbf{J}^*$ indicates the ground-truth coordinates and λ is the hyperparameter that weights each distinct iteration. In our experiments, we set $\lambda = 0.8$ by default.

4. Experiments

4.1. Experiment Settings

We conducted experiments on an NVIDIA A100 GPU with PyTorch. For training, we used the AdamW optimizer [17] with beta1 = 0.5, beta2 = 0.999, and learning rate α = 0.001. The number of input points to the network was sampled to 1,024 and the batch size was set to 32. Batch normalization [14] was applied only in the local encoder and initializer. Meanwhile, to avoid overfitting, we adopted online data augmentation with random rotation ([-120.0, 120.0] degrees around z-axis, [-30.0, 30.0] degrees around x- and y-axis), 3D scaling ([0.8, 1.2]), and 3D translation ([-20, 20] mm). We trained the model for 160 epochs with a learning rate decay of 0.1 after 120 epochs.

4.2. Datasets and Evaluation Metrics

MSRA Dataset. The MSRA dataset [27] provides more than 76K frames captured by the Intel's Creative Interactive

Methods		Mean joint error (mm)			Input	# Params	# FLOPs	Time (ms)
		ICVL	MSRA	NYU				
Ren-4x6x6 [13]		7.63	-	13.39	2D image	-	-	-
Ren-9x6x6 [13]		7.31	9.7	12.69	2D image	-	-	-
Pose-Ren [1]		6.79	8.65	11.81	2D image	-	-	-
DenseReg [32]		7.3	7.2	10.2	2D image	5.8M	-	-
CrossInfoNet [6]		6.73	7.86	10.08	2D image	23.8M	-	-
JGR-P2O [8]		6.02	7.55	8.29	2D image	1.4M	-	2.0 + 7.0
SSRN-SD [24]		6.04	7.16	7.47	2D image	16.2M	1.52G	0.4 + 3.0
SSRN [24]		6.01	7.05	7.37	2D image	22.5M	1.96G	0.4 + 3.4
PHG [23]		5.97	6.94	7.39	2D image	35.7M	27.02G	0.4 + 15.8
3DCNN [11]		-	9.6	14.1	3D voxels	104.9M	-	2.9 + 1.68
SHPR-Net [2]		7.22	7.76	10.78	3D points	-	-	-
HandPointNet [9]		6.94	8.5	10.54	3D points	2.5M	1.72G	8.2 + 11.3
Point-to-Point [12]		6.3	7.7	9.10	3D points	4.3M	2.98G	8.2 + 15.7
V2V [18]		6.28	7.59	8.42	3D voxels	34.1M	6.23G	23 + 5.5
HandFolding [4]		5.95	7.34	8.58	3D points	1.3M	1.31G	8.2 + 3.7
Deng *et al*. [5]		6.05	7.00	**6.84**	3D points	-	-	0.9 + 47.6
Ours	I_{in} =1	6.51	6.93	8.00	3D points	**1.3M**	**0.72G**	0.9 + 4.2
	I_{in} =2	6.12	6.49	7.50	3D points	**1.3M**	1.06G	0.9 + 6.0
	I_{in} =3	5.86	**6.42**	7.36	3D points	**1.3M**	1.40G	0.9 + 7.9
	I_{in} =4	**5.70**	6.48	7.29	3D points	**1.3M**	1.74G	0.9 + 9.7
	I_{in} =5	5.71	6.54	7.27	3D points	**1.3M**	2.08G	0.9 + 11.5

Table 1. Comparison of the proposed method with previous state-of-the-art methods on the ICVL, MSRA and NYU datasets. Input indicates the input type of 2D depth image, 3D voxels or 3D point cloud. # Params indicates the number of model parameters. Time stands for the total computation time including preprocessing time and model inference time.

Gesture Camera. Each frame is annotated with $J = 21$ joints, including one joint for the wrist and four joints for each finger. We followed the leave-one-subject-out cross-validation strategy as used in the previous works [4, 27] to evaluate this dataset.

ICVL Dataset. The ICVL dataset [28] provides 22K training and 1.6K testing depth frames captured by the Intel's Creative Interactive Gesture Camera. Each frame is annotated with $J = 16$ joints, including one joint for the palm and three joints for each finger. We cropped the hand area from a depth image using the method given in [19] because the human body area is present in the frames.

NYU Dataset. The NYU dataset is captured from three different views. Each view contains 72K training 8K testing depth images captured with the PrimeSense 3D sensor. Following recent works [4, 12, 9], we selected one view and 14 joints out of total of 36 annotated joints for training and testing [31]. We also followed the same hand area cropping process [19] as in the ICVL dataset.

Evaluation metrics. We employed two commonly used metrics, the mean joint error and the success rate, to evaluate the performance of hand pose estimation. The mean joint error quantifies the average Euclidean distance between the estimated joint locations and ground-truth ones for all the joints over the entire testing set. The success rate indicates the percentage of the good frames with a mean joint error of less than a given distance.

4.3. Comparison with State-of-the-Art Methods

We compared HandR^{2}N^2 with other state-of-the-art methods, including methods with 2D depth images as input: region ensemble network (Ren-4x6x6 [13], Ren-9x6x6 [32]), Pose-Ren [1], dense regression network (DenseReg) [32], CrossInfoNet [6], JGR-P2O [8], spatial-aware stacked regression network (SSRN [24], SSRN-SD [24]) and pose-guided hierarchical graph network (PHG) [23], and methods with 3D point cloud or voxels as input: 3DCNN [11], SHPR-Net [2], HandPointNet [9], Point-to-Point [12], V2V [18], HandFoldingNet [4] and Deng *et al*. [5].

Table 1 summarizes the performance in terms of the mean joint error. Note that, the evaluations of our model were trained with $I_{tr} = 3$ and tested with $I_{in} = 4$, $I_{in} = 3$, $I_{in} = 5$ on ICVL, MSRA, and NYU, respectively. The results show that HandR^{2}N^2 achieves the new state-of-the-art performance with mean distance errors of 5.70mm, 6.42mm and 7.27mm on two challenging datasets, ICVL and MSRA, respectively. It also achieves the second-lowest error on the NYU dataset. In particular, HandR^{2}N^2 significantly outperforms 3D input based previous state-of-the-arts with large margins. Meanwhile, HandR^{2}N^2 requires

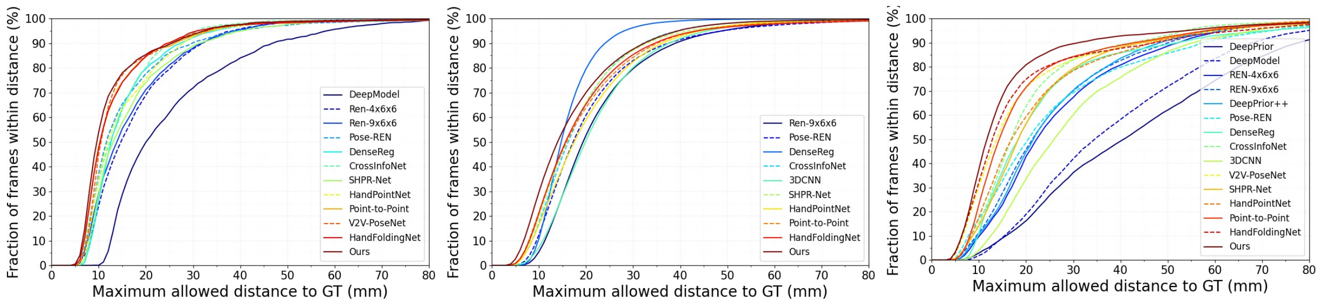

Figure 6. Comparison with the state-of-the-art methods using the ICVL (left), MSRA (middle) and NYU (right) dataset. The success rate is shown in this figure.

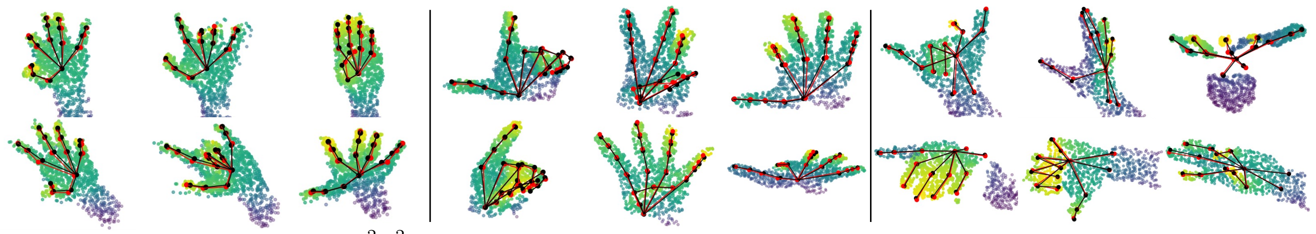

Figure 7. Qualitative results of HandR^{2}N^2 on the ICVL (left), MSRA (middle) and NYU (right) dataset. Hand depth images are transformed into 3D points as shown in the figure. Ground truth is shown in black, and the estimated joint coordinates are shown in red.

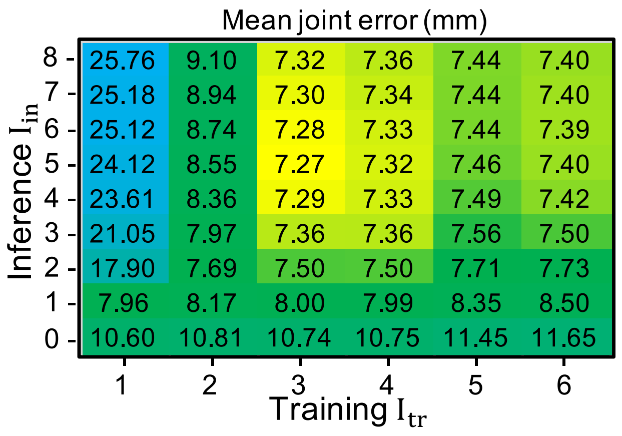

Figure 8. Training I_{tr} vs. inference I_{in}. For each column, the model are trained based on a fixed iteration number and evaluated at different inference I_{in}.

only 1.3M parameters, which is generally less compared to other recent state-of-the-art methods. Figure 6 represents that our method significantly outperforms other methods in terms of success rate when the error threshold is lower than 20mm, 15mm and 52mm on the ICVL, MSRA and NYU datasets, respectively. Figure 2 also illustrates the superior efficiency of HandR^{2}N^2 with dynamically adjustable computation of 0.72+0.34$\times I$ GFLOPs. The results show that our model achieves the lowest estimation error with a similar operation count at $I_{in} = 3$ compared to other state-of-the-art methods. Furthermore, the model can dynamically run more iterations with modestly increased computation for lower estimation error.

4.4. Ablation Study

We conducted ablation experiments to analyze the contribution of each component in our model. The following experiments are evaluated based on the NYU dataset, since it is challenging and far from saturation.

Number of iterations. Although the number of iterations I is a specific number during training, it can be quite flexible and even larger for inference. We first trained the model using different number of training I_{tr} to find the optimal setting of iterations for the training phase. Then we test the trained models with a various inference I_{in}. The results are plotted on Figure 8. Interestingly, when the training $I_{tr} \geq 3$, our model keeps improving for a few more iterations even though the inference I_{in} is larger than the training. In particular, when setting $I_{tr} = 3$ during training, the model achieves the minimal estimation error of 7.27 mm after 5 inference iterations. Please note that, in small number of training iterations ($I_{tr} < 3$), the model is optimized solely for the same inference iterations ($I_{in} = I_{tr}$). On the other hand, larger I_{tr} allows the model to generalize the regularity of changes between all iteration, which benefits not only trained iterations ($I_{in} = I_{tr}$) but also untrained larger iterations ($I_{in} > I_{tr}$). Nonetheless, Figure 8 further demonstrates the reduced performance of our model when $I_{tr} \geq 5$ during training. It is because the lengthy path of back-propagation restricts the model from obtaining appropriate estimations. The longer the iteration path is, the

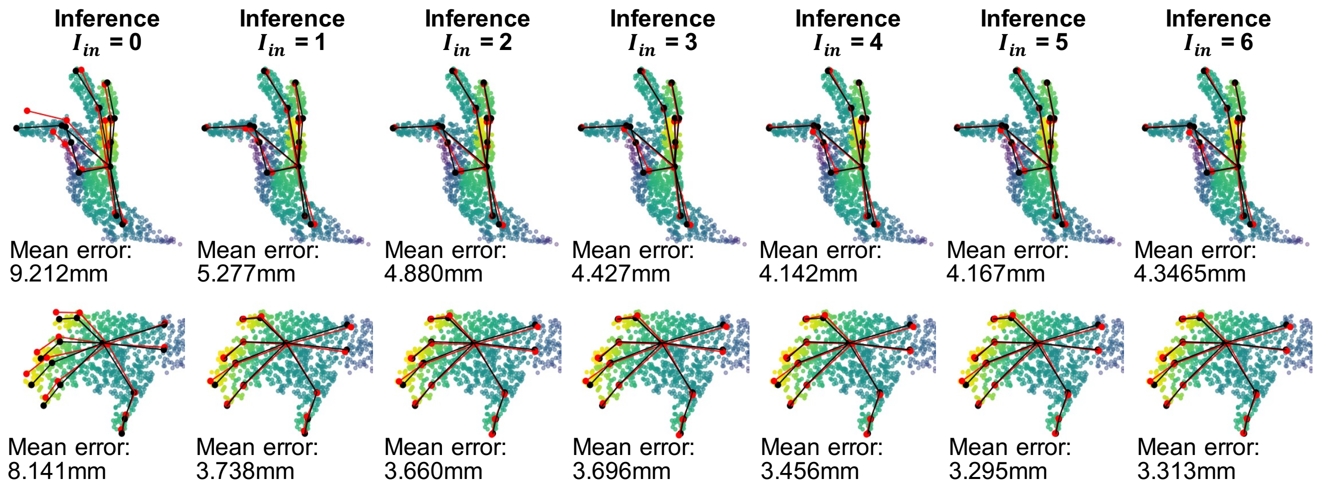

Figure 9. Qualitative visualization of the models with training $I_{tr} = 3$ evaluated at different inference I_{in} based on the NYU dataset. Mean error indicates the mean joint error. Ground truth is shown in black, and the estimated joint coordinates are shown in red.

more different the estimations of the 0-th iteration and the I-th iteration are. Thus, we assume that the accumulated gradients of all iterations to the shared parameters would smooth the right descent gradients for each individual iteration, and thus degrade the final accuracy. However, these smooth accumulated gradients within a specific range of iteration count ($3 < I_{tr} < 5$ according to Figure 8) allow the model to learn the general optimization direction of joints between iterations. Based on the results, we can empirically find the optimal I_{tr} as 3 for high inference performance and training efficiency.

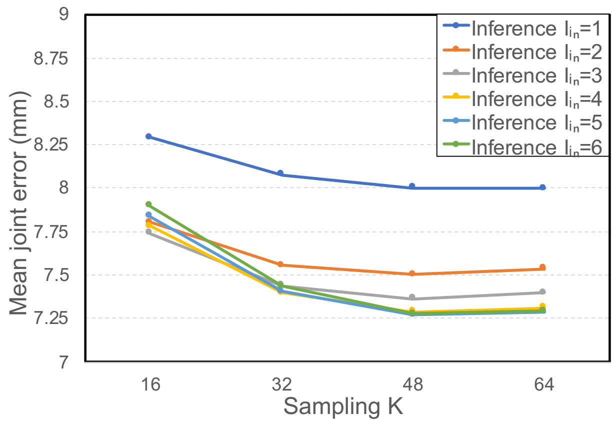

Figure 10. Affect of sampling K. For each line, the inference I_{in}s are same, but with increased K of resampled points for each iteration. All models are trained based on training $I_{tr} = 3$.

In addition, as Table 1 reports, the estimation accuracy is improved with a small number of additional computations (0.34GFLOPs per iteration) and time consumption (1.8 ms per iteration).

Number of resampled points. The resampled points provide K new candidates for location refinement around each joint. Thus, the number K is a significant hyperparameter that affects the performance. Figure 10 presents the performance under different K values and inference iterations. The result shows that larger K reduces the estimation error since it provides a wider range of observation. However, the performance is saturated at $K = 48$, because additionally sampled points beyond the effective region of a particular joint (i.e., points that are distant to the joint and closer to other joints) do not contribute to the estimation.

GRU	GCN	Res.	Mean error
×	×	×	8.68 mm
√	×	×	8.06 mm
×	√	√	7.76 mm
√	√	×	7.62 mm
√	×	√	7.99 mm
√	√	√	7.36 mm

Table 2. Ablations of different GRU configurations. GCN and Res. indicate whether the GCN and residual mechanism are activated in the RRU, respectively. All the ablation models are trained and tested with both $I_{tr} = I_{in} = 3$.

Configurations	Mean error
Conventional GCN + Res.	8.14 mm
Channel-wise GCN + Res.	7.76 mm
Conventional GRU + Res.	8.59 mm
Attentive GRU + Res.	7.99 mm
Conventional GRU + GCN + Res.	8.07 mm
Attentive GRU + Channel-wise GCN + Res.	7.36 mm

Table 3. Comparison with conventional components. All models are trained and tested with both $I_{tr} = I_{in} = 3$.

Analysis of different GRU configurations. To augment the final estimation, we combine the novel GRU with the GCN that evolves the hidden states and residual mechanism for relative representation, as described in Sec. 3. To verify their contributions, we incrementally adopted the GRU cell, the GCN and residual mechanism. Thus, we conducted five ablations as follows: 1) no GRU, 2) GRU; 3) GCN with residual; 4) GRU with GCN; 5) GRU with residual; 6) GRU with both residual and GRU, which is our full configuration. Note that, the ablations without residual, 2) and 4), directly transform the hidden states into absolute joint coordinates. As shown in Table 2, the GRU cell, the GCN and residual mechanism all contribute considerably to the effective estimation. In particular, the novel GRU improves the accuracy by more than 0.4 mm. Furthermore, using the novel GCN and the residual mechanism further reduces the estimation error by more than 0.4mm and 0.2 mm, respectively.

Comparison with the conventional components. To validate the effectiveness of our proposed components, we replace the proposed channel-wise GCN and attentive GRU with the conventional ones, respectively. The implementation details are provided in the supplementary material. Table 3 indicates our proposed GCN and GRU both significantly outperforms the conventional implementations.

5. Conclusion

This paper presented HandR^{2}N^2, a novel recurrent architecture that is capable of iteratively regressing progressively accurate 3D hand pose from a single point cloud. Experimental results showed that our network significantly outperforms previous state-of-the-art methods on three challenging datasets. Extensive experiments also demonstrated the excellent flexibility and efficiency of our network. As limitations, the performance of HandR^{2}N^2 may degrade when hand parts are severely occluded, because the RRU can not sample efficient points. Extending this work to temporal-sequential learning may help to alleviate these limitations, which we leave for future studies.

Acknowledgement.

This work was supported by the Ministry of Science and ICT (MSIT) of Korea under the National Research Foundation (NRF) grant (2022R1A4A3032913) and the Institute of Information and Communication Technology Planning Evaluation (IITP) grants (IITP-2019-0-00421, IITP-2023-2020-0-01821, IITP-2021-0-02052, IITP-2021-0-02068), and by the Technology Innovation Program (RS-2023-00235718) funded by the Ministry of Trade, Industry & Energy (1415187474). Wencan Cheng was partly supported by the China Scholarship Council (CSC).

References

[1] 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, 2, 6

[2] 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

[3] 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. 5

[4] Wencan Cheng, Jae Hyun Park, and Jong Hwan Ko. Handfoldingnet: A 3d hand pose estimation network using multiscale-feature guided folding of a 2d hand skeleton. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11260–11269, 2021. 1, 2, 3, 5, 6

[5] Xiaoming Deng, Dexin Zuo, Yinda Zhang, Zhaopeng Cui, Jian Cheng, Ping Tan, Liang Chang, Marc Pollefeys, Sean Fanello, and Hongan Wang. Recurrent 3d hand pose estimation using cascaded pose-guided 3d alignments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2, 3, 6

[6] Kuo Du, Xiangbo Lin, Yi Sun, and Xiaohong Ma. Crossinfonet: 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, 2, 6

[7] Zhipeng Fan, Jun Liu, and Yao Wang. Adaptive computationally efficient network for monocular 3d hand pose estimation. In *European Conference on Computer Vision*, pages 127–144. Springer, 2020. 3

[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. 1, 2, 4, 6

[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, 6

[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, 2

[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, 2, 6

[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, 6

[13] 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, 2, 6

[14] 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. 5

[15] 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

[16] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 529–537, 2019. 3, 4

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[18] 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, 2, 6

[19] 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

[20] 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, 2

[21] 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. 1, 2, 3

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

[23] Pengfei Ren, Haifeng Sun, Jiachang Hao, Qi Qi, Jingyu Wang, and Jianxin Liao. Pose-guided hierarchical graph reasoning for 3-d hand pose estimation from a single depth image. *IEEE Transactions on Cybernetics*, 2021. 1, 2, 4, 6

[24] Pengfei Ren, Haifeng Sun, Weiting Huang, Jiachang Hao, Daixuan Cheng, Qi Qi, Jingyu Wang, and Jianxin Liao. Spatial-aware stacked regression network for real-time 3d hand pose estimation. *Neurocomputing*, 437:42–57, 2021. 6

[25] 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, 2, 5

[26] Oindrila Saha, Aditya Kusupati, Harsha Vardhan Simhadri, Manik Varma, and Prateek Jain. Rnnpool: efficient nonlinear pooling for ram constrained inference. *Advances in Neural Information Processing Systems*, 33:20473–20484, 2020. 2

[27] 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, 5, 6

[28] 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

[29] 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

[30] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 2

[31] 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, 6

[32] 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

[33] Yiming Wu, Wei Ji, Xi Li, Gang Wang, Jianwei Yin, and Fei Wu. Context-aware deep spatiotemporal network for hand pose estimation from depth images. *IEEE transactions on cybernetics*, 50(2):787–797, 2018. 3

[34] John Yang, Yash Bhalgat, Simyung Chang, Fatih Porikli, and Nojun Kwak. Dynamic iterative refinement for efficient 3d hand pose estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1869–1879, 2022. 2, 3

[35] John Yang, Hyung Jin Chang, Seungeui Lee, and Nojun Kwak. Seqhand: Rgb-sequence-based 3d hand pose and shape estimation. In *European Conference on Computer Vision*, pages 122–139. Springer, 2020. 2, 3

[36] Cheol-Hwan Yoo, Seowon Ji, Yong-Goo Shin, Seung-Wook Kim, and Sung-Jea Ko. Fast and accurate 3d hand pose estimation via recurrent neural network for capturing hand articulations. *IEEE Access*, 8:114010–114019, 2020. 3