Joint-Aware Regression: Rethinking Regression-Based Method for 3D Hand Pose Estimation

Xiaozheng Zheng[12]
zhengxiaozheng@bupt.edu.cn
Pengfei Ren[12]
rpf@bupt.edu.cn
Haifeng Sun[12]
hfsun@bupt.edu.cn
Jingyu Wang[12]
wangjingyu@bupt.edu.cn
Qi Qi[12]
qiqi8266@bupt.edu.cn
Jianxin Liao[12]
liaojx@bupt.edu.cn

[1] State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China

[2] EBUPT Information Technology Co., Ltd.
Beijing, China

Abstract

3D hand pose estimation approaches can be divided into two categories, including regression-based methods and detection-based methods. Detection-based methods utilize fully convolutional networks to obtain hand-crafted coordinate representations like heatmaps and then use a coordinate decoding function like soft-argmax to decode coordinates. In contrast, regression-based methods employ low-dimension features from convolutional networks as unconstrained coordinate representations and then use fully-connected layers to decode coordinates. This way allows the network to learn the coordinate representations and the corresponding coordinate decoding function automatically. However, it causes either weak coordinate representational power or decoding's optimization difficulty. These drawbacks cause regression-based methods far less accurate than detection-based methods. However, detection-based methods require many computations for deconvolution, and their hand-crafted coordinate representations may not be optimal. This paper proposes a novel framework for regression-based methods that can preserve the strength of representations and avoid severe optimization difficulty while remaining flexible, lightweight, and efficient. More specifically, we use joint-specific feature maps as coordinate representations and the joint-shared coordinate decoding module. Moreover, we apply a multi-head mechanism to exploit different coordinate representations and design a learnable re-parameterization method to do multi-stage refinement better. Our approach outperforms state-of-the-art methods on four public benchmarks, including FreiHAND, HO-3D, RHD, and STB.

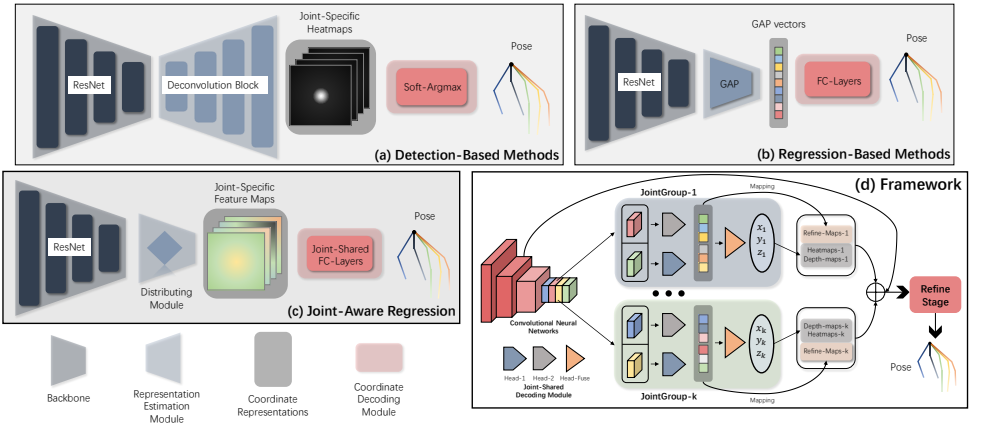

Figure 1: The framework of approaches for hand pose estimation can be divided into three parts, including backbone, representation estimation module, and coordinate decoding module. Detection-based methods (a), regression-based methods (b), and our proposed Joint-Aware Regression (c) all follow this structure. Joint-Aware Regression first utilizes convolutional networks to obtain feature maps. After that, it equably divides feature maps for every joint in channel order. Then, it uses a multi-head joint-shared decoding module to decode joints' coordinates with the corresponding joints' features. Finally, it applies multi-stage refinement with pose re-parameterization to obtain more accurate results.

1 Introduction

Hand pose estimation (HPE) is a widely studied research topic in the field of computer vision, which has various applications in augmented reality (AR), virtual reality (VR), and many other computer vision tasks [7]. Because monocular RGB cameras are much more common and cheap to acquire than multi-camera setups and depth cameras, 3D HPE from a monocular RGB image has become increasingly prevalent in recent years [1, 10, 15, 18, 27, 28, 30, 33, 34, 37]. However, it is still very challenging to achieve efficient and robust estimation due to highly complex backgrounds, large variations in hand pose, occlusions, and fingers' self-similarity.

The approaches of HPE can be categorized into detection-based methods [14, 15, 18, 20, 28] and regression-based methods [10, 25, 26, 27, 30, 33, 34]. As shown in Figure 1 (a) and (b), the framework of those approaches can be divided into three parts, including 1) *backbone* for feature extraction, 2) *representation estimation module* to map feature maps to the desired coordinate representations, and 3) *coordinate decoding module* to decode coordinates from coordinate representations. The main differences between regression-based methods and detection-based methods are 1) *coordinate representations* and 2) *coordinate decoding module*. Detection-based methods utilize fully convolutional networks to predict hand-crafted coordinates representation like heatmaps and use a fixed joint-shared function like soft-argmax to decode the coordinates. They are very accurate due to the good cooperation between heatmaps representations and soft-argmax decoding function. On the one hand, convolutional networks (CNNs) can preserve and exploit the 2D structure information of heatmaps. On the other hand, using soft-argmax function prevents the network from

learning a non-linear function to map 2D coordinate representations to coordinates. There are many works aiming at designing better coordinate representations [8, 15, 20, 21, 24, 31, 35] and better coordinate decoding methods [22, 29, 35]. Tompson et al. [31] first propose heatmaps as coordinate representations. Many works adopt 3D coordinate representations for its accuracy in 3D pose estimation task [8, 21, 24]. Iqbal et al. [15] utilize 2D heatmaps and root-relative depth-maps for 3D HPE. Zhang et al. [35] propose using unbiased heatmaps for more accurate prediction. Newell et al. [22] propose to introduce the second maximal to decode more accurate coordinates. Sun et al. [29] utilize differentiable coordinate decoding function soft-argmax instead of argmax. Zhang et al. [35] explore the distribution structure of the predicted heatmap to infer the underlying maximum activation.

On the contrary, regression-based methods do not use hand-crafted coordinate representations or fixed coordinate decoding functions. Instead, they allow the network to learn coordinate representations and the corresponding coordinate decoding module automatically. The most popular regression-based methods utilize CNNs and global average pooling to obtain low-dimension coordinate representations. Then, they use a coordinate decoding module consisting fully-connected layers to map coordinate representations to coordinates. This way requires much fewer computations than detection-based methods and can avoid severe optimization difficulty caused by using all the feature maps to regress in an overlarge regression space. However, it discards too much spatial structure information due to global average pooling, constraining the regression space too much. This drawback makes it still much less accurate than detection-based methods. Recent popular research about regression-based methods is using Variational Autoencoder (VAE) to encode input images into a small latent space and then recover pose from it [10, 27, 30, 33, 34]. This way puts strong prior in the coordinate representations and the regression space of the coordinate decoding module.

From the discussion above, we can see that coordinate representations and the corresponding coordinate decoding module significantly impact the accuracy and efficiency of HPE. Though detection-based methods have shown that well-designed coordinate representations and coordinate decoding functions can achieve good performance, hand-crafted coordinate representations with fixed coordinate decoding modules are not optimal for all cases. However, it is challenging to design a network that can automatically exploit the best coordinate representations and the corresponding coordinate decoding module because we must simultaneously consider the coordinate representations' ability and the coordinate decoding module's optimization difficulty. From our observation, there are four critical differences between detection-based methods and regression-based methods. Detection-based methods 1) utilize coordinate representations with 2D spatial information, 2) use joint-specific coordinate representations, 3) apply the same coordinate decoding function for all the joints, and 4) exploit different kinds of coordinate representations (e.g., 2D heatmaps and depth-maps [15]). Inspired by those differences, we propose a novel framework for regression-based methods, Joint-Aware Regression (JAR). JAR adopts four strategies corresponding to each difference mentioned above. First, we use *feature maps as coordinate representations* instead of global average pooling features, which can preserve the 2D spatial structure information to maintain the strength of coordinate representations. Second, we divide all the feature maps into *joint-specific feature maps* for coordinate decoding. This way significantly reduces the overlarge regression space caused by using feature maps as coordinate representations. It allows the coordinate decoding module to focus on the corresponding joint sub-space to avoid irrelevant features' disturbance. Third, we utilize a *joint-shared coordinate decoding module*. By this means, we further constrain those joint sub-spaces by guiding them to exploit similar features and reduce the parameters significantly. Last, we additionally employ

a *multi-head joint-shared coordinate decoding module*. In this way, the network can exploit different coordinate representations and combine them to obtain more accurate results. With all these designs, JAR further explores the potential of regression-based methods for HPE while remaining flexible, lightweight, and efficient.

Recent works [25, 26] have shown that regression-based methods can also perform multi-stage refinement like detection-based methods [15, 22, 32] by pose re-parameterization. However, previous pose re-parameterization methods [25, 26] only re-parameterize hand-crafted coordinate representations (e.g., 2D heatmaps, depth maps, or 3D heatmaps). Therefore, it can not fully leverage previous stages' information for refinement. Thus, in our framework, we design a more appropriate refinement strategy for regression-based methods. More specifically, we propose a *learnable re-parameterization method* by mapping joint-specific multi-head outputs to joint-specific refine maps. After that, we concatenate these learnable refine maps, 2D heatmaps, and depth-maps together for refinement. In this way, the subsequent stages can flexibly leverage multi-joint 3D spatial context and powerful dis-ambiguation clues in re-parameterized coordinate representations to boost the performance.

We evaluate JAR on four publicly available 3D hand pose datasets, including FreiHAND [38], HO-3D [11], RHD [37], and STB [36]. Our method outperforms all previous state-of-the-art approaches on these four datasets. In contrast to all previous works, we instead investigate the issues of coordinate representations and coordinate decoding modules of regression-based methods, a largely ignored perspective in the literature. We show that regression-based methods can achieve strong performance in HPE with appropriate architecture design. Crucially, our work is orthogonal to previous works and can easily be applied in other regression-based methods, providing different ideas for future works.

Our main contributions are summarized as follows: 1) We propose a novel regression-based methods framework for 3D hand pose estimation from a monocular RGB image. As far as we know, we are the first to show that regression-based methods can achieve strong performance in hand pose estimation with appropriate design; 2) We propose using joint-specific feature maps as coordinate representations, utilizing multi-head joint-shared coordinate decoding module, and applying learnable multi-stage refinement. These designs significantly improve our framework; 3) We conduct extensive experiments to demonstrate our method's effectiveness. Moreover, our method achieves state-of-the-art performance on FreiHAND, HO-3D, RHD, and STB datasets.

2 Method

2.1 A General Framework for Hand Pose Estimation

As shown in Figure 1, HPE frameworks can be divided into three parts, 1) *backbone* for feature extraction, 2) *representation estimation module* for mapping features to the desired coordinate representations, and 3) *coordinate decoding module* to decode coordinates from coordinate representations. They can be defined as:

$$Y = \mathcal{D}(\mathcal{G}(\mathcal{F}(I))) \tag{1}$$

where $Y \in \mathbb{R}^{k\times 3}$ is the predicted k joints' 3D locations, $I \in \mathbb{R}^{3\times h_i\times w_i}$ is the input RGB image, $\mathcal{F}(\cdot)$ is the backbone, $X = \mathcal{F}(I) \in \mathbb{R}^{c\times h\times w}$ is the feature maps extracted by the backbone, $\mathcal{G}(\cdot)$ is the representations estimation module, and $\mathcal{D}(\cdot)$ is the coordinate decoding module.

Regression-Based Methods. Regression-based methods use CNNs as $\mathcal{F}(\cdot)$ to encode input images into feature maps. Generally, there are two choices of $\mathcal{G}(\cdot)$. One is identity mapping that means directly using X as coordinate representations for decoding. We note this way Feature Regression (FR). The other is global average pooling to squeeze 2D feature maps X into a 1D scalar feature vector $\overline{X}$. We note this way Average Feature Regression (AFR). $\mathcal{D}(\cdot)$ often consists of fully-connected layers and activation layers.

Directly using all the feature maps for decoding like FR will cause severe optimization difficulty of coordinate decoding module $\mathcal{D}(\cdot)$ due to overlarge regression space for regressing from $X \in \mathbb{R}^{c\times h\times w}$. For AFR, on the one hand, using global average pooling features $\overline{X} \in \mathbb{R}^{c\times 1\times 1}$ for decoding can significantly relieve optimization difficulty. On the other hand, the coordinate representations' power is greatly reduced as the spatial structure information is squeezed with global average pooling.

Detection-Based Methods. Detection-based methods also use CNNs as $\mathcal{F}(\cdot)$ to encode input images into feature maps. They further use deconvolutional networks as $\mathcal{G}(\cdot)$ to recover low-resolution feature maps to high-resolution heatmaps. Their $\mathcal{D}(\cdot)$ is a coordinate decoding function like soft-argmax to decode coordinates from heatmaps.

There are two reasons for the high accuracy of detection-based methods. One is they only need to focus on predicting fixed coordinate representations like heatmaps because they utilize a fixed decoding function as $\mathcal{D}(\cdot)$ instead of using a learnable module to fit a function. The other is deconvolutional networks are well suited for reconstructing features with spatial structure information like heatmaps. However, they also have two drawbacks. First, they require lots of computations for deconvolution, as shown in Table 2. Second, they restrict the networks' ability to some extent due to using hand-crafted coordinate representations.

2.2 Joint-Aware Regression

Joint-Aware Regression is designed to learn the best coordinate representations and the corresponding coordinate decoding module automatically. Figure 1 (d) depicts the detailed framework of JAR. We will introduce each part of JAR detailedly in the following.

Joint-Specific Feature Maps. To better preserve the 2D spatial structure information to enhance the power of coordinate representations, we use feature maps as coordinate representations instead of global average pooling features. However, using all the feature maps for regression leads to overlarge regression space. Therefore, we add a prior to constraint the regression space by obtaining the joint-specific feature maps for every joint's decoding. We use a distributing module as $\mathcal{G}(\cdot)$ that equably distributes several feature maps from the last layer of the backbone to a joint in channel order:

$$G_i = \mathcal{G}_i(X) = \mathcal{R}(Concat(X^{id}, X^{id+1}, ..., X^{id+d-1})), i \in \{0, ..., k-1\} \tag{2}$$

where $X^i \in \mathbb{R}^{1\times h\times w}$ is the i^{th} channel of the feature maps $X \in \mathbb{R}^{c\times h\times w}$, $d = \frac{c}{k}$ is the number of feature maps distributed to a joint, $\mathcal{R}$ denotes the reshape operation to flatten 2D feature maps to 1D vector, and $G_i \in \mathbb{R}^{dhw}$ is the obtained coordinate representations of the i^{th} joint.

This way poses a prior that the coordinate decoding module only needs to focus on a feature sub-space of a joint instead of the whole hand feature space. Therefore, the coordinate decoding module's optimization difficulty can be significantly relieved.

Joint-Shared Coordinate Decoding Module. We further add another prior to constraint the

regression space by using a joint-shared coordinate decoding module. It is defined as:

$$Y_i = \mathcal{D}^s(G_i) = \mathcal{T}(G_i) = ReLU(G_i W_1 + b_1)W_2 + b_2 \tag{3}$$

where $\mathcal{D}^s$ denotes coordinate decoding for single head, Y_i is the i^{th} joint's 3D coordinates, W_i and b_i denotes the learnable parameters of linear transformations layer, and $\mathcal{T}$ denotes the transformations consisting of two linear transformations with a *ReLU* activation in between.

This way can further constrain the regression space because it guides all the joints' sub-spaces to exploit the same features (e.g., guide all sub-spaces similar to heatmaps). It also allows the coordinate decoding module to use all the joints' information to update itself.

Multi-Head Decoding Module. Considering the variety of the potential coordinate representations (e.g., 2D heatmaps, depth maps, or 3D heatmaps), we apply a multi-head mechanism to allow the network to exploit them simultaneously. More specifically, we divide the joint-specific feature maps into multiple parts, making a part responsible for a kind of coordinate representation. Joint-shared coordinate decoding module with multi-head is defined as:

$$Y_i = \mathcal{D}^m(G_i) = \mathcal{T}_f(M_i) \qquad M_i = Concat(\mathcal{T}_0(G_i^0), ..., \mathcal{T}_{n-1}(G_i^{n-1})) \tag{4}$$

where $\mathcal{D}^m$ denotes coordinate decoding for multi-head, n is the head number, G_i^j denotes the j^{th} head's features of the i^{th} joint, $G_i = concat(G_i^0, ..., G_i^{n-1})$, $\mathcal{T}_i$ denotes the transformations for the i^{th} head, M_i denotes the multi-head outputs of the i^{th} joint, and $\mathcal{T}_f$ denotes the transformations for fusing multi-head outputs to regress final coordinates.

With this multi-head mechanism, we can further utilize the feature maps from the backbone to strengthen the coordinate representations' power to exploit many more beneficial representations for the final results automatically.

Learnable Multi-Stage Refinement. Recent works [25, 26] have demonstrated that introducing multi-stage refinement by using pose re-parameterization can significantly boost the performance of regression-based methods. The predicted results can provide rich 3D spatial information and powerful disambiguation clues for further stage networks to refine coarse predictions. However, previous pose re-parameterization methods only re-parameterize hand-crafted coordinate representations (e.g., 2D heatmaps and depth-maps), which can not fully leverage previous stages' information for the refinement of the following stages. Thus, we propose a learnable re-parameterization method to reparameterize the multi-head output features of every joint to joint-specific refine maps for additional guidance:

$$R_i = \mathcal{R}(\mathcal{T}_r(Detach(M_i))) \tag{5}$$

where R_i is the refine maps for the i^{th} joint, $\mathcal{T}_r$ is the transformations for reparameterizing multi-head output features to refine maps, *Detach* operation means stop the gradient flow to previous networks, and $\mathcal{R}$ is reshape operation to transform 1D vector to 2D refine maps.

Apart from learnable reparameterized refine maps, we also generate 2D heatmaps H and depth-maps D based on the previous stage predicted joints for refinement:

$$H_i(p) = exp(\frac{\|p - p_i^{pred}\|}{\sigma^2}), p \in \Omega \qquad D_i(p) = \begin{cases} z_i, & p = p_i^{pred} \\ 0, & otherwise \end{cases} \tag{6}$$

where p_i^{pred} represents predicted 2D coordinates of the i^{th} joint, σ controls the standard deviation of the heatmaps, Ω denotes the set of all pixel locations, and z_i represents the predicted root-relative depth from the previous stage.

ID	Method	EPE(mm) ↓	AUC ↑	△(EPE, AUC)
1	baseline (FR)	14.90	0.918	(+0.00,+0.000)
2	+ global average pooling (AFR)	14.08	0.927	(+0.82, +0.009)
3	+ joint-specific pooling features	13.84	0.938	(+1.06, +0.020)
4	+ joint-shared decoding	12.53	0.945	(+2.37,+0.027)
5	+ joint-specific feature maps	12.21	0.948	(+2.69,+0.030)
6	+ joint-shared decoding	12.10	0.949	(+2.80,+0.031)
7	+ multi-head decoding	11.68	0.954	(+3.22,+0.036)
8	+ heatmaps refine	11.30	0.957	(+3.60,+0.039)
9	+ depth-maps refine	11.19	0.958	(+3.71,+0.040)
10	+ refine-maps refine	10.97	0.961	(+3.93,+0.043)
11	+ third stage refine	10.87	0.961	(+4.03,+0.043)

Table 1: Ablation studies on RHD dataset.

Finally, we concatenate the middle-level feature maps from the first stage CNNs, refine maps, 2D heatmaps, and depth-maps for refinement:

$$A^j = Concat(F^0, R^j, H^j, D^j) \tag{7}$$

where A^j denotes the final features for the refinement of stage $j+1$, F^0 denotes the middle-level feature maps from the first stage CNNs, R^j, H^j, and D^j denote all the joints' refine maps, heatmaps, and detph-maps of stage j, respectively.

3 Experiments

3.1 Datasets and Metrics

FreiHAND [38] is a real-world 3D hand dataset containing 130,240 training images and 3960 testing images from multi-view setups. Images are annotated with mesh and pose labels. It is challenging because the backgrounds between training and testing images vary greatly. The evaluation is performed at an online server.

HO-3D [11] is a real-world 3D hand dataset containing hand-object interaction images. The dataset is made of 68 sequences, totaling 77,558 frames of 10 users manipulating one among ten different objects. It contains 66,034 training images and 11,524 testing images. All the images are with mesh and pose annotations. The evaluation is performed at an online server.

RHD [37] is a synthetic dataset of rendered hand images. It contains 41,258 training images and 2728 testing images, including 20 different characters doing 39 different actions. The training set and testing set share no same character or action. Every RGB image of this dataset has corresponding pose labels and mask labels.

STB [36] is a real-world dataset containing 18,000 images. All the images are with pose labels. It consists of twelve sequences of data from video under different lighting conditions and six different backgrounds. Following previous works [2, 37], we modify the palm joint to the wrist joint and use ten sequences for training and another two sequences for testing.

Evaluation Metrics. To evaluate the performance of our method, we use the two most common metrics, mean end-point-error (EPE) and area under the curve (AUC) of the percentage

	R-18	R-34	R-50	R-101	R-152	FLOPs(G)	Params(M)
2.5D (Det)	11.92	11.56	11.43	11.04	10.69	12.9(7.6)	34.0(10.5)
AFR (Reg)	14.08	13.38	12.46	12.25	12.14	5.6(0.2)	32.0(8.5)
JAR (Reg)	11.68	10.86	10.74	10.40	10.16	5.6(0.2)	28.9(5.4)

Table 2: Comparisons between different methods with different backbones. Det denotes detection-based methods, Reg denotes regression-based methods, and the number in the brackets denotes the FLOPs and params of the network apart from the backbone.

of correct keypoints (PCK). EPE is calculated as the average Euclidean distance between the predicted pose and ground-truth pose. AUC is computed as the percentage of predicted joints errors falling within certain error thresholds compared with ground-truth joints.

3.2 Implementation Details

Training. We train and evaluate our method on a computer with an Intel(R) Core(TM) i9-10940X CPU@3.30GHz CPU, 64GB of RAM, and an Nvidia GeForce RTX 3090 GPU having 24GB of GPU memory. The operating system of the computer is Ubuntu 18.04.3 LTS. All the experiments are implemented with PyTorch-1.7.1 framework [23]. We use ResNet [13] pretrained on ImageNet [6] as our backbone. The batch size is set to 64. The model is optimized with Adam solver [16] with an initial learning rate $3e-4$. The learning rate is decayed with *cosine* schedule. Our model is trained 40 epochs on FreiHAND and 30 epochs on HO-3D, RHD, and STB. We use $L1$ loss for all the stages.

Data Processing. We crop the hand from the original image according to the bounding box and resize the cropped image to 256×256. On FreiHAND, we crop the hand from the center, do not use given ground-truth hand scale information, and use root depth provided by I2L-MeshNet [20] for a fair comparison. On HO-3D, we use the 2D keypoints information as bounding box during training, rely on the provided bounding box during testing, and do not use scale information, following [4]. On RHD and STB, following previous work [18, 34], we use the ground-truth hand masks to generate bounding box. Moreover, we use the ground-truth root depth and scale information, following [2, 9, 15, 18, 27, 30, 33, 34, 37]. Data augmentation, including scaling (±10%), rotation (±180°), translation (±10 pixels), and color jittering (±20%) are performed during training.

3.3 Results

Ablation studies. To understand the influence of the individual parts of our method, we add them to the baseline method one at a time and evaluate the performance. We use Equation 3 for FR, AFR, every head, and multi-head fuse. Besides, we try to make the number of their parameters at the same level for fair comparisons. We use 8 feature maps for a head and 8 heads (experiments for tuning these numbers are shown in Supp. material).

As shown in Table 1, FR as the baseline performs the worst (ID-1). Because AFR reduces the regression space using global average pooling, it achieves better performance (ID-2) and requires fewer computations and parameters. There is a dramatic performance improvement when using joint-specific feature maps instead of all the feature maps for decoding (2.69mm, comparing ID-1 and ID-5). This phenomenon shows the benefits of dividing the whole

Method	mesh	scale	FPS	FreiHAND		HO-3D
				EPE	AUC	EPE*
CVPR19-Boukhayma et al.[1]	✓	✓	/	35.0	0.351	/
CVPR19-Hasson et al.[12]	✓	✓	/	13.3	0.737	31.8
ICCV20-MANO CNN [33]	✓	✓	/	10.9	0.783	/
CVPR20-Hampali et al. [11]	✓	/	/	/	/	30.4
CVPR20-YoutubeHand [17]	✓	/	/	8.4	0.834	/
ECCV20-I2L-Mesh [5]	✓	/	53	7.4	/	/
ECCV20-Pose2Mesh [20]	✓	/	8	7.4	/	/
CVPR21-Chen et al. [3]	✓	/	64	6.9	0.863	/
CVPR21-MeshTransformer [19]	✓	/	18	6.5	/	/
CVPR21-Chen et al. [4]	/	/	/	11.8	0.77	/
AAAI21-Li et al. [18]	/	✓	48	8.6	0.842	/
Ours(ResNet-34,1-stage)	/	/	**134**	7.3	0.856	26.9
Ours(ResNet-34,2-stage)	/	/	86	6.8	0.865	25.7
Ours(ResNet-34,4-stage)	/	/	49	**6.5**	**0.870**	**25.1**

Table 3: Comparisons on FreiHAND and HO-3D. * means with scale and trans alignment.

feature space into joint-specific sub-spaces to constrain the regression space. Moreover, after using a joint-shared coordinate decoding module instead of joint-specific ones to constrain the regression space further, the performance gains another 0.11mm improvement (ID-6). Besides, this way reduces the decoding module's parameters by twenty times.

Experiments show that adding feature maps for a head leads to saturated improvements (Supp. material). However, applying the multi-head mechanism to exploit more features can improve the performance significantly (0.42mm, ID-7). This result shows the benefits of allowing the network to exploit different coordinate representations. Furthermore, those priors can also bring improvements to AFR (ID-3 and ID-4). Especially, the joint-shared coordinate decoding module improves AFR a lot (ID-4). Here, ID-4 and ID-6 use the same dimension of coordinate representations, but ID-6 performs much better. This phenomenon proves the importance of using coordinate representations with spatial structure information.

Last, we show the effect of multi-stage refinement. From our experiments (Supp. material), we find that only using the last block of ResNet for refinement stages to extract features can achieve good performance with our framework. Therefore, we adopt this scheme to avoid multi-stage refinement introducing too many computations. Only using 2D heatmaps for refinement can decrease 0.38mm error (ID-8). Further incorporating depth-maps for refinement can bring another 0.11mm improvement (ID-9). Finally, by incorporating learnable refine maps, the performance gains another 0.22mm improvement (ID-10). This result demonstrates that only reparameterizing hand-crafted features can not fully leverage previous stages for refinement, and using learnable refine maps can provide much more information. When adding stage number to 3, the improvement is limited on RHD (0.1mm, ID-11).

Comparisons with two baselines using different backbones. Table 2 uses RHD dataset to compare the performance of the most common regression-based method (AFR), the recent popular detection-based method for 3D HPE [15, 18, 28] (2.5D), and JAR. The comparisons between 2.5D and AFR show a performance gap between regression-based methods and detection-based methods. However, after adding priors on regression-based methods, JAR

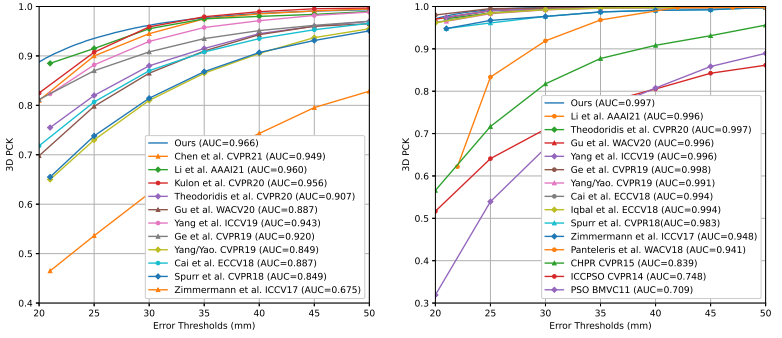

Figure 2: Comparisons of 3D PCK on RHD (left) and STB (right) datasets.

significantly improves the performance and surpasses 2.5D with all the backbones. This result proves that allowing the network to learn the coordinate representations and the corresponding decoding module can achieve better performance if the regression-based methods are well-designed. On the last two columns of Table 2, we show the FLOPs and parameters of them with ResNet-50. Due to the heavy deconvolutional blocks (7.6G-FLOPs and 10.5M-Params), 2.5D is much less efficient. In contrast, JAR retains the advantages of regression-based methods and uses the least FLOPs and params while achieves the best performance.

3.4 Comparisons with State-of-the-arts

FreiHAND and HO-3D. We compare with state-of-the-art methods [1, 3, 4, 5, 11, 12, 18, 19, 20, 38] on FreiHAND and HO-3D datasets. As shown in Table 3, our method with ResNet-34 as the backbone and four stages can achieve the best performance on these two datasets without using mesh labels for training or ground-truth hand scale information. Furthermore, we test the inference speed of these methods with their publicly released code on our computer. Our method has the highest inference speed for the same performance.

RHD and STB. We use ResNet-34 as the backbone and two stages for our model to compare with state-of-the-art methods [2, 3, 9, 10, 15, 17, 18, 27, 30, 33, 34, 37] on RHD and STB datasets in AUC metric. The comparisons are shown in Figure 2. We outperform all other methods on RHD dataset and achieve comparable performance on STB dataset with other methods due to its saturation.

4 Conclusions

In this paper, we propose a novel regression-based methods framework, Joint-Aware Regression. This framework preserves the strength of coordinate representations and simultaneously relieves the coordinate decoding module's optimization difficulty. We achieve this by using joint-specific feature maps as coordinate representations, a multi-head joint-shared module for coordinate decoding, and a learnable multi-stage refinement. We show that regression-based methods can achieve strong performance with appropriate design while remaining lightweight and efficient for hand pose estimation. Extensive experiments on public benchmarks demonstrate the effectiveness and efficiency of our method.

References

[1] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 3d hand shape and pose from images in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10843–10852, 2019.

[2] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–682, 2018.

[3] Xingyu Chen, Yufeng Liu, Chongyang Ma, Jianlong Chang, Huayan Wang, Tian Chen, Xiaoyan Guo, Pengfei Wan, and Wen Zheng. Camera-space hand mesh recovery via semantic aggregation and adaptive 2d-1d registration. *arXiv preprint arXiv:2103.02845*, 2021.

[4] Yujin Chen, Zhigang Tu, Di Kang, Linchao Bao, Ying Zhang, Xuefei Zhe, Ruizhi Chen, and Junsong Yuan. Model-based 3d hand reconstruction via self-supervised learning. *arXiv preprint arXiv:2103.11703*, 2021.

[5] Hongsuk Choi, Gyeongsik Moon, and Kyoung Mu Lee. Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose. In *European Conference on Computer Vision*, pages 769–787. Springer, 2020.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[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.

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

[9] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 3d hand shape and pose estimation from a single rgb image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 10833–10842, 2019.

[10] Jiajun Gu, Zhiyong Wang, Wanli Ouyang, Jiafeng Li, Li Zhuo, et al. 3d hand pose estimation with disentangled cross-modal latent space. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 391–400, 2020.

[11] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3196–3206, 2020.

[12] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11807–11816, 2019.

[13] 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.

[14] Weiting Huang, Pengfei Ren, Jingyu Wang, Qi Qi, and Haifeng Sun. Awr: Adaptive weighting regression for 3d hand pose estimation. In *AAAI*, pages 11061–11068, 2020.

[15] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5 d heatmap regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 118–134, 2018.

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

[17] Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael M Bronstein, and Stefanos Zafeiriou. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4990–5000, 2020.

[18] Moran Li, Yuan Gao, and Nong Sang. Exploiting learnable joint groups for hand pose estimation. *arXiv preprint arXiv:2012.09496*, 2020.

[19] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. *arXiv preprint arXiv:2012.09760*, 2020.

[20] Gyeongsik Moon and Kyoung Mu Lee. I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. *arXiv preprint arXiv:2008.03713*, 2020.

[21] 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.

[22] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[23] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[24] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7025–7034, 2017.

[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.

[26] 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.

[27] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. Cross-modal deep variational hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–98, 2018.

[28] Adrian Spurr, Umar Iqbal, Pavlo Molchanov, Otmar Hilliges, and Jan Kautz. Weakly supervised 3d hand pose estimation via biomechanical constraints. *arXiv preprint arXiv:2003.09282*, 2020.

[29] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 529–545, 2018.

[30] Thomas Theodoridis, Theocharis Chatzis, Vassilios Solachidis, Kosmas Dimitropoulos, and Petros Daras. Cross-modal variational alignment of latent spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 960–961, 2020.

[31] Jonathan Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *arXiv preprint arXiv:1406.2984*, 2014.

[32] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[33] Linlin Yang and Angela Yao. Disentangling latent hands for image synthesis and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9877–9886, 2019.

[34] Linlin Yang, Shile Li, Dongheui Lee, and Angela Yao. Aligning latent spaces for 3d hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2335–2343, 2019.

[35] Feng Zhang, Xiatian Zhu, Hanbin Dai, Mao Ye, and Ce Zhu. Distribution-aware coordinate representation for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7093–7102, 2020.

[36] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 3d hand pose tracking and estimation using stereo matching. *arXiv preprint arXiv:1610.07214*, 2016.

[37] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *Proceedings of the IEEE international conference on computer vision*, pages 4903–4911, 2017.

[38] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 813–822, 2019.