

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

Xiaozheng Zheng¹²
zhengxiaozheng@bupt.edu.cn

Pengfei Ren¹²
rpf@bupt.edu.cn

Haifeng Sun¹²
hfsun@bupt.edu.cn

Jingyu Wang¹²
wangjingyu@bupt.edu.cn

Qi Qi¹²
qiqi8266@bupt.edu.cn

Jianxin Liao¹²
liaojx@bupt.edu.cn

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

² EBUP Information Technology Co., Ltd.
Beijing, China

Abstract

Our supplementary materials are organized as follows. Section 1 introduces the experiments of applying JAR in other regression-based methods; Section 2 provides more ablation studies; Section 3 presents qualitative results.

1 Application in Other Methods

We implement two recent regression-based methods (RBMs) [4, 10] for 3D hand pose estimation (HPE). After that, we apply Joint-Aware Regression (JAR) in them to show our method’s versatility. As shown in Table 1, JAR brings significant improvements to both of these methods. Next, we will introduce the details of these experiments.

1.1 VAE-based methods

VAE-based methods [10, 4, 4, 4, 10] use convolutional networks (CNNs) to encode the input RGB image into a latent space and then use fully-connected layers to decode the sampled variables from the latent space to recover the final pose. We adopt the baseline in [10] as our baseline model. The details of this baseline model are shown in the left of Table 2. The baseline model uses ResNet-18 as the backbone (BB) to encode RGB images to feature maps, global average pooling (GAP) with two fully-connected layers as representation estimation module (REM) to encode features to latent space, and four fully-connected layers with one

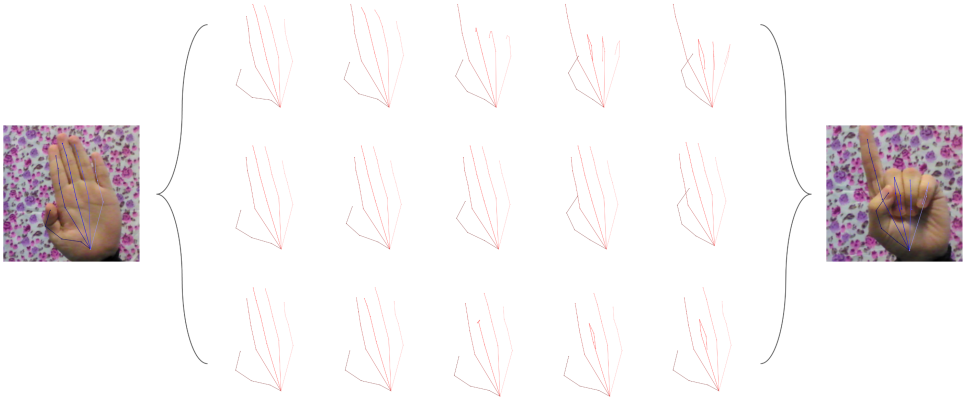


Figure 1: Latent space interpolation. The far left and far right show two input images and their predicted pose. The middle shows the interpolated pose generated from linear interpolations on the latent space. The first row shows the whole latent space interpolation results. Due to our method dividing whole latent space into several joint-specific spaces, we can interpolate between joints we want, which is more flexible in generations. The second row and the third row show thumb and middle finger latent space interpolations, respectively.

dropout layer as the coordinates decoding module (CDM) to decode 3D pose. After that, we use JAR with one head to replace the REM of CDM in the baseline model. The detailed network is shown in the right of Table 2. Instead of encoding GAP features into a whole latent space, this modified model encodes joint-specific feature maps into joint-specific latent spaces responsible for the corresponding joints. Then, it decodes every joint’s location with samples from the corresponding latent space.

The experiments are conducted on RHD dataset [10]. We utilize the same training scheme as our primary contents do. The dimension of the latent variable is set to 64. For more materials about VAE-based methods, we recommend referring to [10]. From the results in Table 1, we can see that JAR brings 3.39mm (21%) significant improvements to VAE-based methods. Apart from performance improvement, VAE-based methods with JAR disentangle the latent space from the whole pose space into a joint-wise space, which increases the flexibility of hand generation. Figure 1 illustrates the flexible interpolations in latent space.

1.2 SRN

Stacked Regression Network (SRN) [9] is an accurate and lightweight model for 3D HPE from a depth image. It uses a differentiable re-parameterization module to reconstruct 3D heatmaps and unit vector fields from predicted joint coordinates, allowing stack multiple regression modules to improve performance.

This experiment is evaluated on the public depth images dataset, NYU dataset [8]. We utilize ResNet-18 as the backbone and JAR with eight heads to replace the traditional regression methods in SRN. The batch size is set to 32. We use the Adam optimizer with an initial learning rate of 1e-3 to train 40 epochs. The learning rate decreases to 1e-4 after 20 epochs. For more details, we recommend referring to [9].

As shown in Table 1, JAR brings significant improvement to SRN with one stage (7.1%).

Method	EPE (mm)	\triangle (mm)
Yang et al. [14]	16.61	0.00
Our implementation	16.53	0.08
+ JAR	13.14	3.47
Ren et al. [15]-1Stage	9.81	0.00
Our implementation	8.99	0.82
+ JAR	8.35	1.46
Ren et al. [15]-2Stage	7.78	0.00
Our implementation	7.89	-0.11
+ JAR	7.52	0.26
+ RM	7.45	0.33

Table 1: Performance of VAE-Based methods and SRN with JAR.

Part	Baseline		+ JAR	
	Operation	Shape	Operation	Shape
BB	Input ResNet-18	(3, 256, 256) (512, 8, 8)	Input ResNet-18	(3, 256, 256) (512, 8, 8)
REM	GAP FC - μ FC - σ	(512) (64) (64)	Conv2D-Reshape FC - μ FC - σ	(21, 512) (21, 64) (21, 64)
CDM	Sample FC-ReLU FC-ReLU FC-ReLU Dropout FC-Reshape	(64) (512) (512) (512) (512) (21, 3)	Sample FC-ReLU FC-ReLU FC-ReLU Dropout FC	(21, 64) (21, 512) (21, 512) (21, 512) (21, 512) (21, 3)

Table 2: Detailed network. Left is VAE-baseline model and right is VAE-JAR model.

When using two stages, the improvements are less obvious than using one stage (4.7%). We think this is because this dataset is saturated. When adding additional refine maps (RM) for refinement, the performance can get another 0.07mm improvement.

2 Ablation Studies

Here, we provide more ablation studies tuning the best choice of 1) residual blocks for refinement stages' backbone, 2) the number of feature maps for a head, and 3) the number of heads.

2.1 Comparisons between different backbone blocks for refinement stage

Previous works [12, 13, 14] usually utilize many residual blocks for refinement stages' backbone. However, this will introduce much more parameters and computations overhead.

Blocks/Performance	1	2	3	4
w/o RM (mm)	11.19	11.14	11.34	11.55
w RM (mm)	10.97	10.96	10.99	10.95
Params (MB)	27.08	29.91	33.57	33.71
FLOPs (GB)	3.2G	3.7G	4.4G	5.0G

Table 3: Comparisons between different numbers of blocks for refinement stage’s backbone.

Moreover, we think the re-parameterized features are high-level features that do not need to fuse with low-level features. Therefore, we only use one block for the refinements stages’ backbone.

As shown in Table 3, only using one block can achieve almost the best performance with better efficiency. Moreover, the performance starts degenerating when using more than two blocks. Besides, we can see that our re-parameterized refine maps bring significant improvements for different numbers of blocks, especially for more blocks. This phenomenon further proves the superiority of introducing refine maps to provide more information for refinement adaptively.

2.2 Comparisons between different feature maps numbers for one head

We conduct experiments to tune the best number of feature maps distributed to one joint. All the experiments are done with one head. The results are shown in Table 4. We find that even with one feature map for a joint, JAR can achieve performance (13.87mm) better than AFR (14.08mm). This phenomenon demonstrates that reserving spatial information to enhance the power of coordinates representations (CR) is essential. When increasing the feature map number, JAR’s performance is even better. However, this improvement is gradually saturated after the number reaches 8. This saturation may be caused by feature redundancy and optimization difficulty caused by more parameters. Thus, we adopt eight feature maps for a head for our final model.

2.3 Comparisons between different head numbers

We also attempt to use different numbers of heads. As shown in Table 5, using eight heads is the best choice. When the head number is small, the network can not fully exploit different representations for robust predictions. As the head number increases, the performance becomes better. When the head number is bigger than 8, more representations become redundant.

2.4 Comparisons between different multi-head method

We also try another two kinds of multi-head methods. The first one is similar to transformer [8]. More specifically, this way uses the same features with different parameters for decoding. In other words, this is using the same kind of coordinates representations with multiple different coordinate decoding modules. The second one is the opposite of the first one, which uses different features with the same parameters for decoding. This way works more like an

Feature Maps Num	1	2	4	8	16	32
Performance	13.87	12.88	12.28	12.08	12.03	12.06

Table 4: Comparisons between different feature maps numbers for a head.

Head Num	1	2	4	8	16
Performance	12.08	11.94	11.73	11.68	11.69

Table 5: Comparisons between different head numbers.

ensemble method for allowing the network to learn multiple similar representations to obtain more robust results.

As shown in Table 6, both of these two methods perform worse than using different coordinates representations and different coordinates decoding (CD) modules simultaneously, especially when using large backbones. This phenomenon shows the importance of using CR with the corresponding specific CD module.

3 Qualitative Results

We present some visual comparisons between different methods and the visual results of JAR on four different datasets.

3.1 Comparisons between different methods

Figure 6 presents some qualitative comparisons between Average Feature Regression (AFR), Latent 2.5D [14] (L25D), and JAR on RHD testing images. From the results, we can see that JAR performs much better with complex poses, severe occlusions, and poor lighting conditions. Moreover, JAR can predict more accurate depth values to obtain 3D hand poses.

3.2 Visual results on different datasets

We present some visual results on FreiHAND (Figure 2), HO-3D (Figure 3), RHD (Figure 4), and STB (Figure 5). The last row of these figures shows some failure cases (except STB).

On the FreiHAND dataset, we can see that JAR can solve extreme viewpoints, hand-object interactions, and complex poses. We can see from the failure cases that JAR has difficulty solving particularly complex poses (fingers cross each other) and hands with extreme scales. On the HO-3D dataset, JAR obtains good results with objects’ occlusions. However, particularly severe occlusions will lead to totally wrong results. On the RHD dataset, JAR can solve complex poses but have difficulty with extreme lighting conditions. On the STB

	ResNet-18	ResNet-50
Same CR / Diff CD	11.81	11.10
Diff CR / Same CD	11.77	10.96
Diff CR / Diff CD	11.68	10.74

Table 6: Comparisons between different multi-head methods.

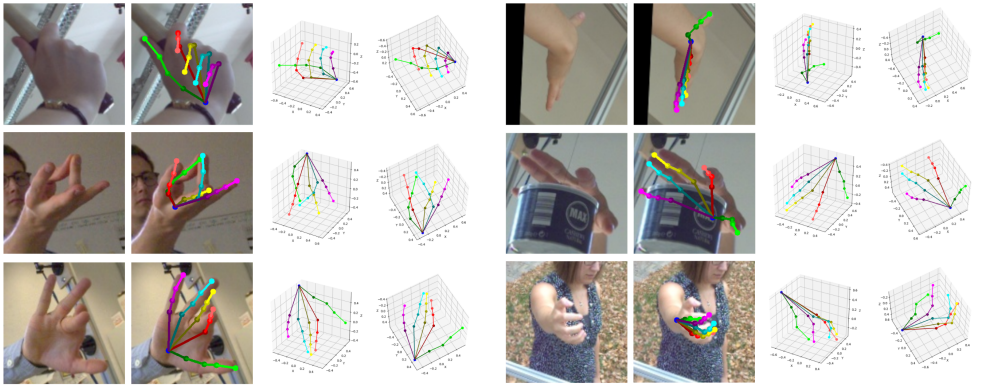


Figure 2: Visual results on FreiHAND dataset. The last row shows some failure cases.

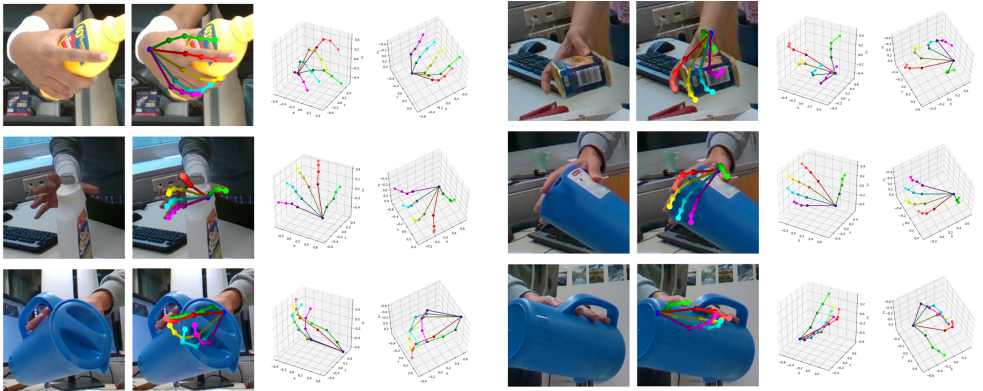


Figure 3: Visual results on HO-3D dataset. The last row shows some failure cases.

dataset, JAR can also solve the unusual poses between two gestures change. Because STB is relatively easier, we do not observe severe failed cases.

References

- [1] 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.
- [2] 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.
- [3] 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.
- [4] Pengfei Ren, Haifeng Sun, Weiting Huang, Jiachang Hao, Daixuan Cheng, Qi Qi,

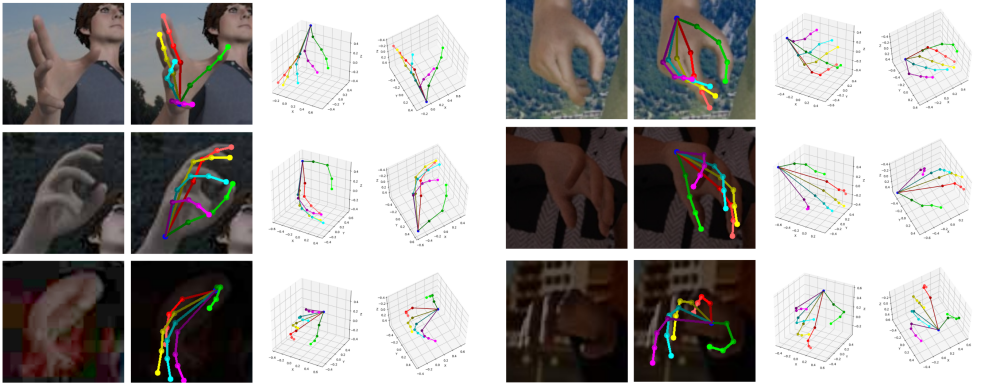


Figure 4: Visual results on RHD dataset. The last row shows some failure cases.

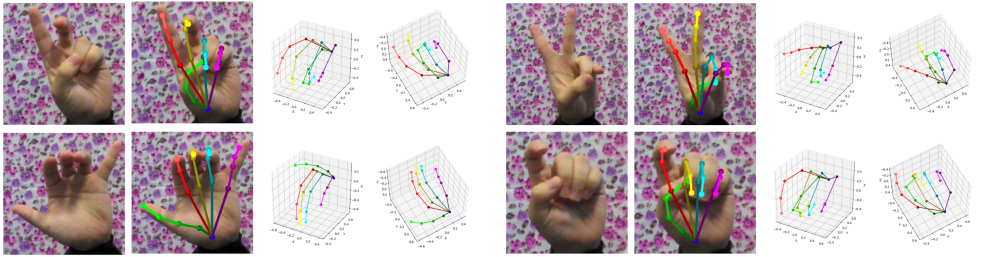


Figure 5: Visual results on STB dataset.

Jingyu Wang, and Jianxin Liao. Spatial-aware stacked regression network for real-time 3d hand pose estimation. *Neurocomputing*, 437:42–57, 2021.

- [5] 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.
- [6] 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.
- [7] 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.
- [8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [9] 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.

- [10] 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.
- [11] 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.

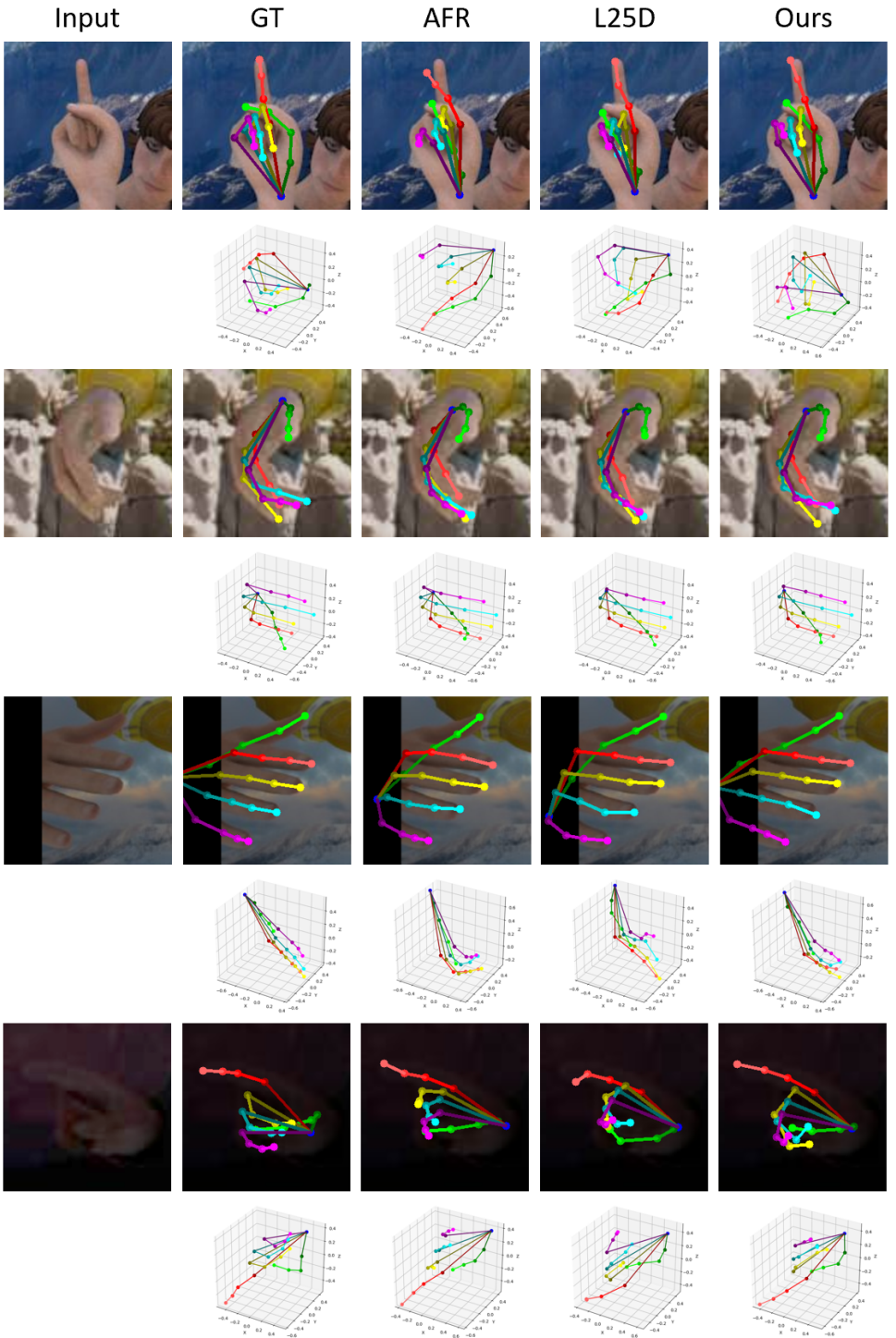


Figure 6: Visual results between different methods.