Aligning Latent Spaces for 3D Hand Pose Estimation Supplementary Material

Linlin Yang*[1], Shile Li*[2], Dongheui Lee[2,3], Angela Yao[4]
*Equal contribution
[1]University of Bonn, Germany [2] Technical University of Munich, Germany
[3]German Aerospace Center, Germany [4]National University of Singapore, Singapore

Section A provides the formulation of Viewpoint Correction in the main manuscripts; Section B provides the details of our network architectures. More results and ablation studies are presented in Section C and Section D respectively. The use of corresponding data is discussed in Section E. Note that all the notation and abbreviations here are consistent with the main manuscript.

A. Viewpoint Correction

The aim of viewpoint correction is to rotate the 3D hand, such that the 3D centroid of the hand aligns with the camera's z-axis. More specifically, if we assume that the crop's center coordinates on the original image is $[u_c, v_c]$, the rotation matrix $\mathbf{R}_{vc} \in \mathbb{R}^{3\times3}$ to correct for the viewing angle can be obtained as follows:

$$\begin{aligned} \alpha_y &= \text{atan2}(u_c - o_x, f), \\ \widetilde{\mathbf{c}} &= \mathbf{R}_y(-\alpha_y) \cdot [u_c - o_x, v_c - o_y, f]^T, \\ \alpha_x &= atan2(\widetilde{\mathbf{c}}_2, \widetilde{\mathbf{c}}_3), \\ \mathbf{R}_{vc} &= \mathbf{R}_y(-\alpha_y) \cdot \boldsymbol{R}_x(\alpha_x), \end{aligned} \tag{1}$$

whereas f is the camera focal length, o_x, o_y are the camera center coordinates, $\widetilde{\mathbf{c}}$ is an intermediate result and $\alpha_{x,y}$s are rotation angles around corresponding axis. After viewpoint correction, the 3D poses and point clouds are subtracted with the hand's centroid point and normalized to a canonical size.

B. Model Architectures

In this section, we list the detailed parameters for the various encoders and decoders. For encoding RGB images, we use ResNet-18 [2]; for encoding point clouds, we use ResPEL [4]; for decoding heatmaps, we follow the decoder architecture DCGAN [6]; for decoding point clouds, we follow the decoder architecture FoldingNet [9]; for decoding 3D hand poses, we use four fully connected layers with Relu. Specific architecture details of the encoders and decoders are given in Tables 1-5.

Encoder for RGB hand images
Input: RGB image $\mathbf{x}$
ResNet-18
FC-(N64)

Table 1: The encoder architecture for hand RGB images. We use 256×256 RGB images as the input of ResNet-18 [2] and set the dimensionality of latent variable $\mathbf{z}$ to 64.

Encoder for point cloud
Input: Point cloud $\mathbf{w}_1$
ResPEL-(N64, $\times$ 7)
ResPEL-(N256, $\times$ 7)
ResPEL-(N512, $\times$ 7)
Point to hidden unit voting(N128)
FC-(N128), BN, Relu
FC-(N64), Tanh

Table 2: The encoder architecture for point clouds. We use 256×3 point clouds as the input of ResPEL [4] and set the dimensionality of latent variable $\mathbf{z}$ to 64.

Abbreviations: N for number of kernels or neurons, FC stands for fully connected layers, CONV stands for convolutional layers with 1×1 kernels, stride of size 1 and without padding, TCONV stands for transposed convolutional layers with 5×5 kernels, stride of size 2 and padding of 'SAME', BN stands for batch normalization layers. For example, FC-(N512) refers to a fully connected layer with 512 neurons. ResPEL(N64, $\times$ 7) stands for one Residual Permutation Equivariant Layer block [4] with 64 hidden neurons and 7 layers.

C. Additional Results

In this section, we show more results on RHD and STB, and add experiments on Dexter+Object dataset. In Fig. 1,

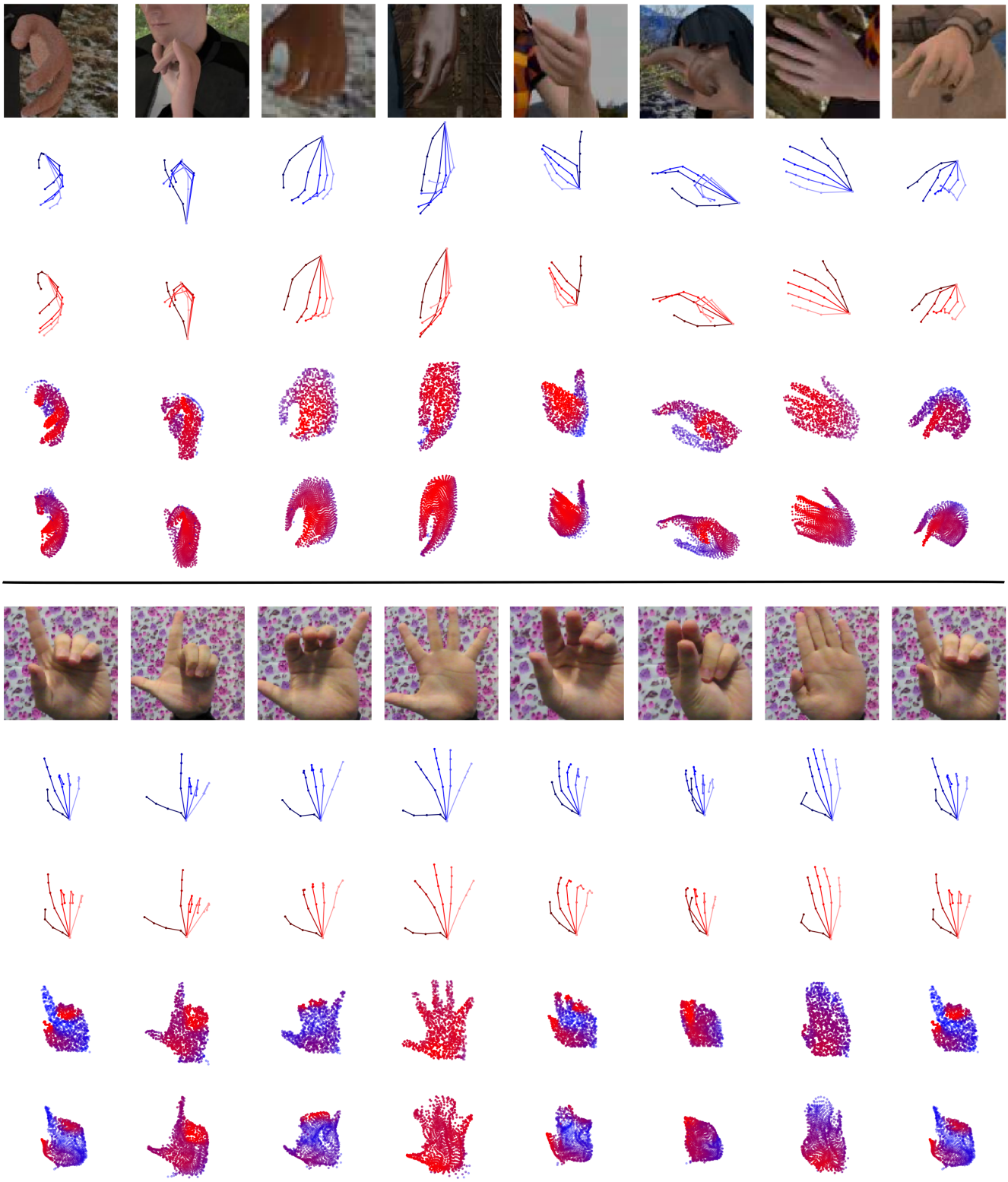

Figure 1: More examples on 3D pose estimation and point cloud reconstruction for RHD (top) and STB (bottom) dataset. From top to bottom: RGB images, ground-truth poses in blue, estimated poses from $\mathbf{z}_{rgb}$ in red, ground-truth point clouds, reconstructed point clouds from $\mathbf{z}_{rgb}$. The color for point clouds decodes the depth information, closer points are more red and further points are more blue. Note that the ground-truth point clouds are not used for inference, it is shown here only for comparison purpose.

Decoder for heatmaps
Input: Latent variable **z**
FC-(N8192)
Reshape(8,8,128), BN
TCONV-(N64), BN, Relu
TCONV-(N32), BN, Relu
TCONV-(N21), Sigmoid

Table 3: The decoder architecture for heatmaps. The final shape is (64,64,21) for heatmaps.

Decoder for point clouds	
Input: Latent variable **z**	
Fold1	Concat(**z**,init) CONV-(N256), BN, Relu CONV-(N512), BN, Relu CONV-(N1024), BN, Relu CONV-(N512), BN, Relu CONV-(N3), BN, Tanh
Output	middleOutput
Fold2	Concat(**z**,middleOutput) CONV-(N256), BN, Relu CONV-(N512), BN, Relu CONV-(N1024), BN, Relu CONV-(N512), BN, Relu CONV-(N3), BN, 3*Tanh

Table 4: The decoder architecture for point clouds with FoldingNet [9]. The final shape is (256,3,1) for point clouds.

Decoder for 3D hand poses
Input: Latent variable **z**
FC-(N128),Relu
FC-(N128),Relu
FC-(N128),Relu
FC-(N63)

Table 5: The decoder architecture for 3D hand poses. The final shape is (21,3,1) for 3D hand poses.

we show some qualitative examples of poses and point clouds decoded from the $\mathbf{z}_{\text{rgb}}$. The 3D poses and point clouds can be successfully reconstructed from the same latent variable **z**. In Fig. 2, we compare the PCK curve of our approach with other state-of-the-art methods [3, 5] on Dexter+Object dataset following [3, 5]. We can see that our method outperform [3, 5]. Note that this comparison is unfair since our method use both the best hand root position and the best global scale here.

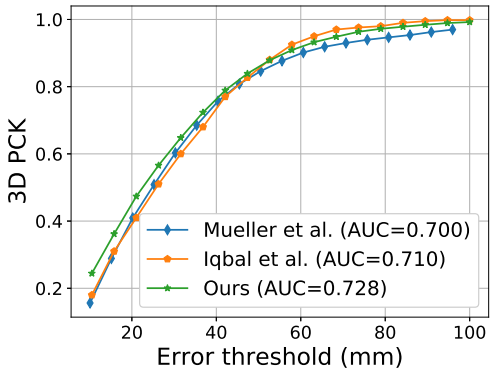

Figure 2: AUC: Comparison to state-of-the-art methods on the Dexter+Object dataset. Ours refers to S4 in Table 1 (RC2CHP).

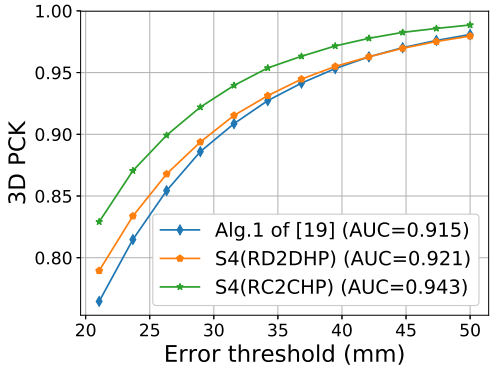

Figure 3: Ablation Study: Our S4, "RC2CHP" and "RD2DHP" vs [7] "RC2CHP" on the RHD dataset.

D. Ablation Study

We prefer point clouds instead of standard depth maps, due to superior performance of using point clouds. To compare the two modalities, we use depth maps (D) instead of point clouds in our S4. The decrease in performance using depth maps vs point clouds is shown in Fig. 3 (orange vs green). For encoding and decoding depth maps, we follow the architecture DCGAN [6]. To show the effectiveness and efficiency of our S4 training strategy, we compare our strategy with [7]'s iterative training strategy. We use the same encoder/decoder network architectures, same modalities (*i.e.* RGB, point clouds and heatmap), same hyperparameters and directly use their iterative training strategy to train "RC2CHP". We show the results in Fig. 3 (green vs blue). We observe that our S4 outperforms [7]'s strategy and also converges faster (our 50 epochs vs. their 600 epochs).

E. Corresponding data

We utilize corresponding data pairs to align the latent space; this is our main contribution. With the aid of corresponding data, our training strategy, using a **joint objective**, can converge much faster and get better joint representations. If S4 uses only non-corresponding data in semi-/weakly-supervised setting, this deteriorates the quality of the latent alignment; pose accuracy is reduced by $\sim$2mm. Corresponding data is generally required for joint objective training and concurrent works have the same requirement [1, 8]. This is not too much of a restriction in practice because there are several existing multi-modal datasets (*e.g.* NYU, SynthHands, HandDB) and synthetic multimodal data is easy to obtain. If corresponding data is limited, different encoders can be pre-trained on large amounts of non-corresponding data separately and then aligned using limited amount of corresponding data. Our semi-/weakly-supervised setting only needs small number of RGBD-pose pairs to align the latent space and the majority of training relies only on RGB-depth pairs (by setting the weighting of the pose estimation loss to zero), which can be easily recorded with any RGBD sensor. Compared to our baseline S1 which is fully supervised, we are more accurate with only 25% of RGBD-pose pairs.

References

[1] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *ECCV*, 2018. 4

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[3] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5 d heatmap regression. In *ECCV*, 2018. 3

[4] Shile Li and Dongheui Lee. Point-to-pose voting based hand pose estimation using residual permutation equivariant layer. In *CVPR*, 2019. 1

[5] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *CVPR*, 2018. 3

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 1, 3

[7] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. Cross-modal deep variational hand pose estimation. In *CVPR*, 2018. 3

[8] Linlin Yang and Angela Yao. Disentangling latent hands for image synthesis and pose estimation. In *CVPR*, 2019. 4

[9] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 1, 3