Supplementary Material for
Extract-and-Adaptation Network for 3D Interacting Hand Mesh Recovery

JoonKyu Park[1*] Daniel Sungho Jung[2,3*] Gyeongsik Moon[4*] Kyoung Mu Lee[1,2,3]
[1]Dept. of ECE&ASRI, [2]IPAI, Seoul National University, Korea
[3]SNU-LG AI Research Center, [4]Meta Reality Labs Research
{jkpark0825, dqj5182}@snu.ac.kr, mks0601@meta.com, kyoungmu@snu.ac.kr

In this supplementary material, we provide more experiments, discussions, and other details that could not be included in the main text due to the lack of pages. The contents are summarized below:

S1. Detailed architecture

In the main manuscript, we introduce the Extract-and-Adaptation Network (EANet) as a means of effectively extracting and processing interactions between two hands. The proposed EANet mainly operates upon the key component, EABlock, to extract and adapt features from the two hands. In this section, we provide a detailed explanation of the EABlock and its primary component, the FuseFormer, which are illustrated in Figures 4 and 5 of the main manuscript, respectively.

S1.1. FuseFormer

In Algorithm S1, we show the inference process of FuseFormer in the extract stage of EABlock. From two hand features ($\mathbf{F}_{\text{L}}$ and $\mathbf{F}_{\text{R}}$), JoinToken $\mathbf{t}_{\text{J}}$ is obtained by passing a concatenated two hand features to a fully connected layer after reshaping. To obtain SimToken $\mathbf{t}_{\text{S}}$, we first reshape each hand feature ($\mathbf{F}_{\text{L}}$ or $\mathbf{F}_{\text{R}}$) and concatenate the reshaped left and right hand features with a class token $\mathbf{t}_{\text{cls}}$. A concatenated token $\mathbf{t}$ is made after reshaping the concatenated feature of $\mathbf{t}_{\text{cls}}$, $\mathbf{F}_{\text{L}}$, and $\mathbf{F}_{\text{R}}$. The concatenated token $\mathbf{t}$ is then used for extracting query $\mathbf{q}_{\text{SA}}$, key $\mathbf{k}_{\text{SA}}$, and value $\mathbf{v}_{\text{SA}}$ with separate linear layers in a self-attention (SA) based Transformer. After preparing two tokens ($\mathbf{t}_{\text{J}}$ and $\mathbf{t}_{\text{S}}$), a cross-attention (CA) based Transformer processes the two novel tokens, JoinToken $\mathbf{t}_{\text{J}}$ as query $\mathbf{q}_{\text{CA}}$ and SimToken $\mathbf{t}_{\text{S}}$ as key-value pair ($\mathbf{k}_{\text{CA}}$ and $\mathbf{v}_{\text{CA}}$), to effectively fuse two hand features ($\mathbf{F}_{\text{L}}$ and $\mathbf{F}_{\text{R}}$) and output an interaction feature $\mathbf{F}_{\text{inter}}$.

S1.2. EABlock

We show the inference process of EABlock in Algorithm S2. From left hand feature $\mathbf{F}_{\text{L}}$ and right hand feature $\mathbf{F}_{\text{R}}$, a FuseFormer in the extract stage of EABlock first extracts an interaction feature $\mathbf{F}_{\text{inter}}$. Then, with the extracted interaction feature $\mathbf{F}_{\text{inter}}$ in the extract stage, each FuseFormer in the adaptation stage of EABlock fuses each hand feature ($\mathbf{F}_{\text{L}}$ or $\mathbf{F}_{\text{R}}$) and the interaction feature $\mathbf{F}_{\text{inter}}$ to produce an adapted interaction feature for each hand ($\mathbf{F}_{\text{interL}}$ or $\mathbf{F}_{\text{interR}}$). Lastly, our EABlock constructs the final outputs ($\mathbf{F}_{\text{L}}^{*}$ and $\mathbf{F}_{\text{R}}^{*}$) after a fully connected layer on each adapted interaction feature ($\mathbf{F}_{\text{interL}}$ or $\mathbf{F}_{\text{interR}}$) and then a concatenation to each hand feature ($\mathbf{F}_{\text{L}}$ or $\mathbf{F}_{\text{R}}$), respectively.

S2. Number of adaptation stages in EABlock

To adapt the extracted interaction feature to each hand feature, our EABlock fuses an interaction feature and one hand feature (*i.e.* left hand feature or right hand feature) to obtain an adapted interaction feature for each hand. Here, we justify our strategy to adapt only once to each hand feature. Table S1 shows that a single iteration of the adaptation stage shows comparable results to two iterations of the adaptation stages. Considering the additional costs followed by additional adaptation stages, we set the number of adaptation stages to 1.

S3. MPJPE comparison with state-of-the-art methods

Table 5 in the main manuscript provides a comparison of MPVPE and MRRPE for various methods, while Table S2 presents additional results based on MPJPE. We follow the

Algorithm S1 Pseudocode of FuseFormer in a PyTorch-style

```
class FuseFormer(nn.Module):
    def __init __():
        FC = nn.Linear(1024, 512)
        t_cls = nn.Parameter(512, 1)
        Q_SA = nn.Linear(512, 512)
        K_SA = nn.Linear(512, 512)
        V_SA = nn.Linear(512, 512)
        Q_CA = nn.Linear(512, 512)
        K_CA = nn.Linear(512, 512)
        V_CA = nn.Linear(512, 512)
        ψ : R^{1024×8×8} → R^{64×1024} # reshape
        φ : R^{512×8×8} → R^{512×64} # reshape
        η : R^{512×64} → R^{129×512} # reshape
        softmax = nn.Softmax(dim=-1)
        MLP = nn.Sequential(
                        nn.Linear(512, 512*4),
                        nn.Linear(512*4, 512))
        MLP′ = nn.Sequential(
                        nn.Linear(512, 512*4),
                        nn.Linear(512*4, 512))
        d_{k_SA}, d_{k_CA} = 512, 512
    def forward(F_L, F_R): # F_L and F_R ∈ R^{512×8×8}
        # get JoinToken
        t_J = FC(ψ(torch.cat((F_L,F_R)))) # ∈ R^{64×512}
        # get SimToken
        F_L = φ(F_L)# ∈ R^{512×64}
        F_R = φ(F_R)# ∈ R^{512×64}
        t = η(torch.cat((t_cls,F_L,F_R))) # ∈ R^{129×512}
        q_SA, k_SA, v_SA = Q_SA(t), K_SA(t), V_SA(t)
        r = softmax((q_SA@k_SA^T)/√d_{k_SA})v_SA + t
        t_S = r + MLP(r)
        # process two tokens
        q_CA = Q_CA(t_J)
        k_CA, v_CA = K_CA(t_S), V_CA(t_S)
        r′ = softmax((q_CA@k_CA^T)/√d_{k_CA})v_CA + t_J
        F_inter = r′ + MLP′(r′)
        return F_inter
```

# of adaptation	MPJPE			MPVPE			MRRPE
	Single	Two	All	Single	Two	All	
0	10.66	11.81	11.25	8.37	11.24	9.61	21.00
1	9.62	**11.54**	**10.58**	**7.86**	10.78	**9.13**	**18.82**
2	**9.60**	11.90	10.75	8.14	**10.66**	9.19	**18.82**

Table S1: **Comparison of models with various number of adaptation stages.**

evaluation protocol of prior studies by using a pre-defined joint regression matrix to obtain the joint positions from the estimated meshes, which are scaled using the ground truth bone length. Note that the MPJPE results for Keypoint Transformer in Table S2 differ from those reported in their main manuscript, as our results are obtained using their official codes and pre-trained weights based on the mesh representation, while their reported numbers were based on a 2.5D representation. Our results demonstrate that EANet outperforms other methods in 3D joint estimation, indicating its effectiveness.

Algorithm S2 Pseudocode of EABlock in a PyTorch-style

```
class EABlock(nn.Module):
    def __init __():
        FuseFormer_extract = FuseFormer()
        FuseFormer_L = FuseFormer()
        FuseFormer_R = FuseFormer()
        FC = nn.Linear(512,128)
    def forward(F_L, F_R): # F_L and F_R ∈ R^{512×8×8}
        # interaction feature extract
        F_inter = FuseFormer_extract(F_L, F_R)
        # interaction feature adaptation
        F_interL = FuseFormer_L(F_L, F_inter)
        F_interR = FuseFormer_R(F_R, F_inter)
        F*_L = torch.cat((F_L, FC(F_interL)))
        F*_R = torch.cat((F_R, FC(F_interR)))
        return F*_L, F*_R
```

Methods	MPJPE		
	Single	Two	All
Zhang *et al.* [8]	-	13.48	-
Keypoint Transformer [2]	11.04	16.22	13.82
IntagHand [3]	-	8.79	-
EANet (Ours)	**5.19**	**6.57**	**5.88**

Table S2: **Quantitative comparison with state-of-the-art methods on InterHand2.6M [4] dataset.**

S4. Measuring the pose difference for Figure 9

In L833-L850 and Figure 9, we present the pose difference between two hands across different sequences of the InterHand2.6M dataset. To compute the pose difference for each selected sequence, we begin by flipping a right hand to a left hand. Next, we subtract the root joint (wrist) position of each hand. Finally, we align the global rotation of the two hands to solely determine the finger pose symmetricity.

S5. Additional qualitative comparisons

In the below figures, we further show the qualitative comparisons with other state-of-the-art methods [3, 2]. Specifically, in Figure S1 and S2, we show visual comparisons on InterHand2.6M [4] dataset. In Figure S3, we show visual comparisons on in-the-wild interacting hand images.

S6. Discussion

S6.1. Limitations

Despite recent progress on domain adaptation [1, 5, 6] that provides several techniques for reducing the discrepancy between training data and test data, our work has not yet employed such designs for better generalization ability. In Figure S3, our EANet shows reasonable results on in-the-wild images without such designs. However, we believe that additional designs to improve generalization ability on in-the-wild images can further improve the performance of EANet in future works.

S6.2. Societal impacts

Our EANet performs robust 3D interacting hand mesh recovery with potential applications on technologies connecting real-world and virtual-world such as AR and VR. Especially, the work is useful in cases where two hands are often interacting, including virtual meetings and virtual events.

License of the Used Assets

- InterHand2.6M dataset [4] is CC-BY-NC 4.0 licensed.
- HIC dataset [7] is publicly available dataset.
- Zhang *et al*. [8] codes are released for academic research only, and it is free to researchers from educational or research institutes for non-commercial purposes.
- Keypoint Transformer [2] codes are released for academic research only, and it is free to researchers from educational or research institutes for non-commercial purposes.
- IntagHand [3] codes are released for academic research only, and it is free to researchers from educational or research institutes for non-commercial purposes.

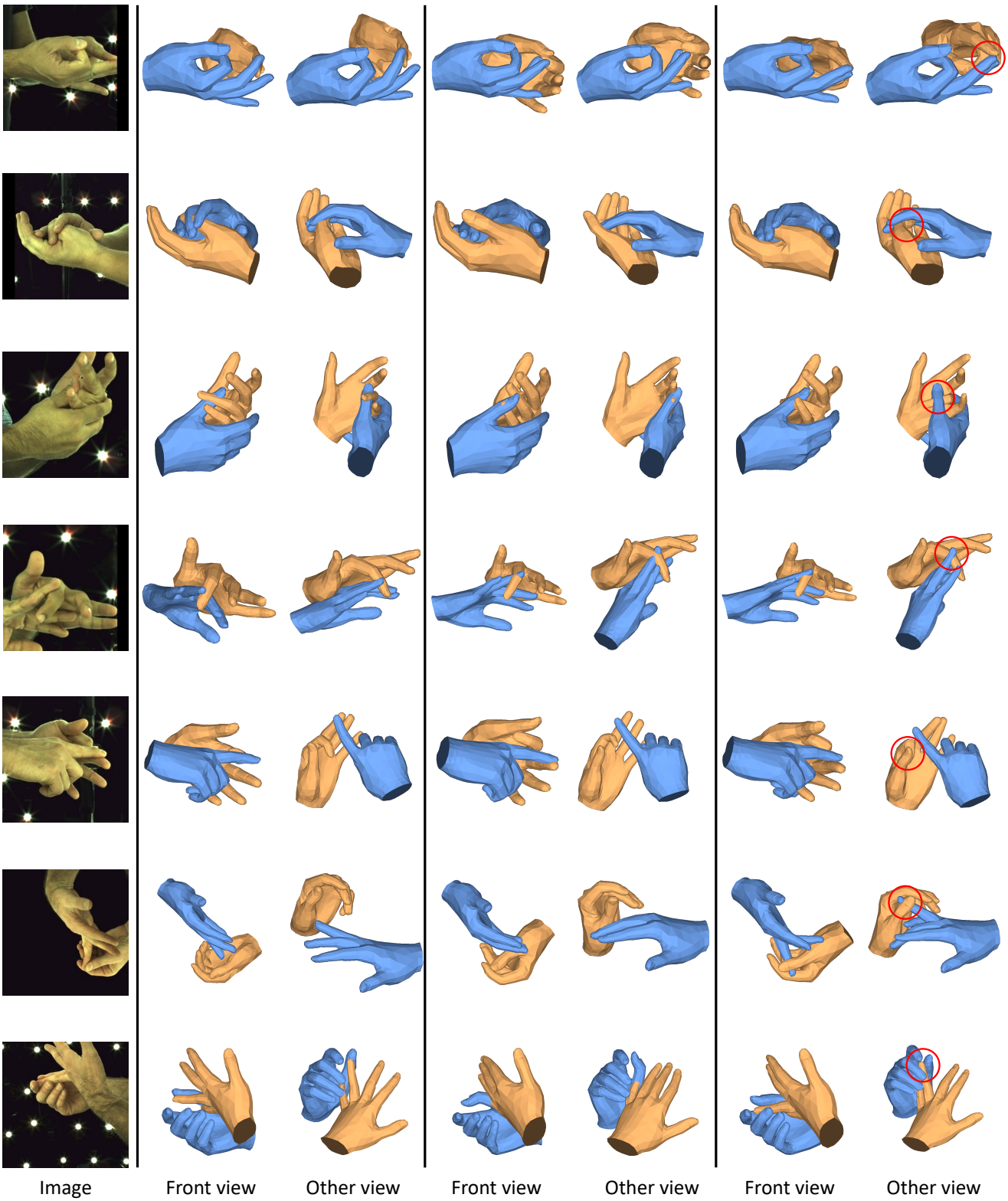

Figure S1: **Visual comparison with state-of-the-art methods on InterHand2.6M [4]. The red circles highlight regions where our EANet is correct, while others are wrong.**

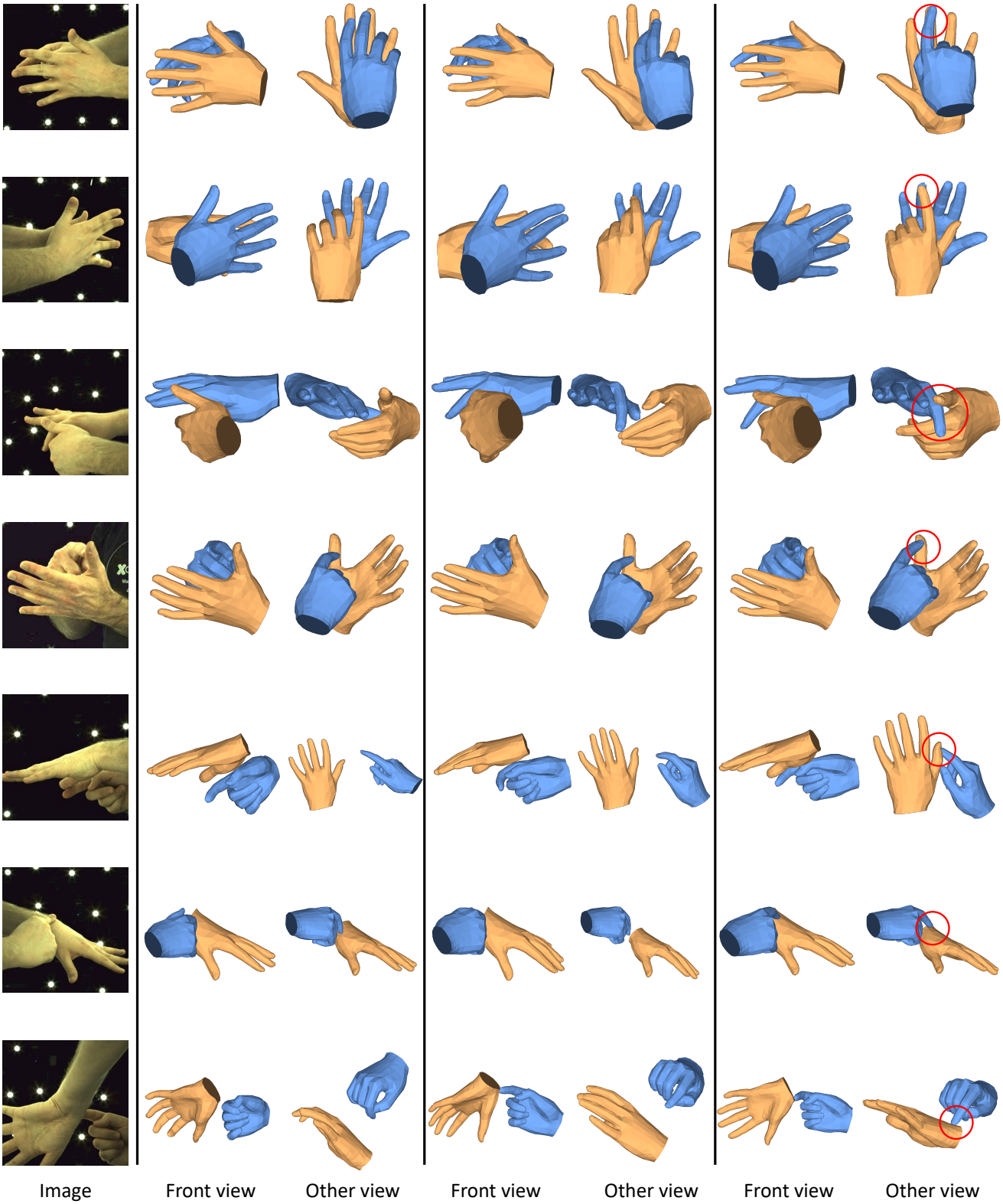

Figure S2: **Visual comparison with state-of-the-art methods on InterHand2.6M [4]. The red circles highlight regions where our EANet is correct, while others are wrong.**

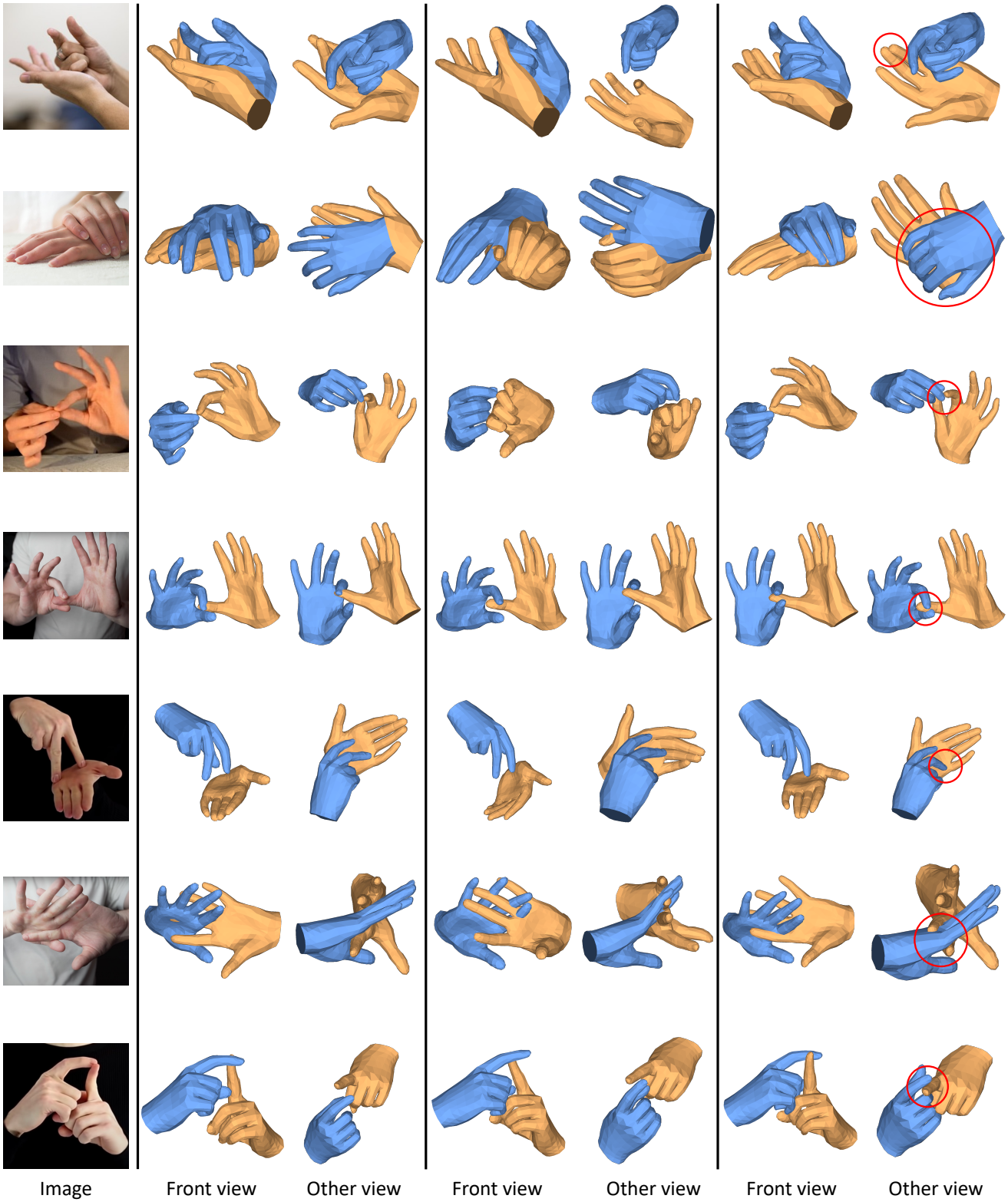

Figure S3: **Visual comparison with state-of-the-art methods on in-the-wild images. The red circles highlight regions where our EANet is correct, while others are wrong.** The images are crawled from Google and YouTube.

References

[1] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 3

[2] Shreyas Hampali, Sayan Deb Sarkar, Mahdi Rad, and Vincent Lepetit. Keypoint Transformer: Solving joint identification in challenging hands and object interactions for accurate 3D pose estimation. In *CVPR*, 2022. 2, 3, 4, 5, 6

[3] Mengcheng Li, Liang An, Hongwen Zhang, Lianpeng Wu, Feng Chen, Tao Yu, and Yebin Liu. Interacting attention graph for single image two-hand reconstruction. In *CVPR*, 2022. 2, 3, 4, 5, 6

[4] Gyeongsik Moon, Shoou-I Yu, He Wen, Takaaki Shiratori, and Kyoung Mu Lee. InterHand2.6M: A dataset and baseline for 3D interacting hand pose estimation from a single RGB image. In *ECCV*, 2020. 2, 3, 4, 5

[5] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *AAAI*, 2018. 3

[6] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017. 3

[7] Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. Capturing hands in action using discriminative salient points and physics simulation. *IJCV*, 2016. 3

[8] Baowen Zhang, Yangang Wang, Xiaoming Deng, Yinda Zhang, Ping Tan, Cuixia Ma, and Hongan Wang. Interacting two-hand 3D pose and shape reconstruction from single color image. In *ICCV*, 2021. 2, 3