This ICCV workshop paper is the Open Access version, provided by the Computer Vision Foundation. Except for this watermark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

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

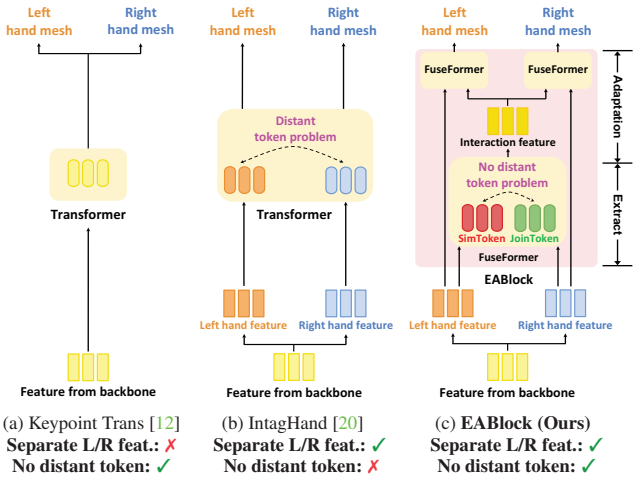
Abstract

Understanding how two hands interact with each other is a key component of accurate 3D interacting hand mesh recovery. However, recent Transformer-based methods struggle to learn the interaction between two hands as they directly utilize two hand features as input tokens, which results in distant token problem. The distant token problem represents that input tokens are in heterogeneous spaces, leading Transformer to fail in capturing correlation between input tokens. Previous Transformer-based methods suffer from the problem especially when poses of two hands are very different as they project features from a backbone to separate left and right hand-dedicated features. We present EANet, extract-and-adaptation network, with EABlock, the main component of our network. Rather than directly utilizing two hand features as input tokens, our EABlock utilizes two complementary types of novel tokens, SimToken and JoinToken, as input tokens. Our two novel tokens are from a combination of separated two hand features; hence, it is much more robust to the distant token problem. Using the two type of tokens, our EABlock effectively extracts interaction feature and adapts it to each hand. The proposed EANet achieves the state-of-the-art performance on 3D interacting hands benchmarks. The codes are available at https://github.com/jkpark0825/EANet.

1. Introduction

Recovering 3D meshes from two interacting hands [27, 39, 18, 10, 12, 20] is necessary for immersive experiences in AR/VR and human-computer interaction. Although the interactions between two hands are highly prevalent in the real world, building a robust 3D mesh recovery framework for two interacting hands is still an open challenge.

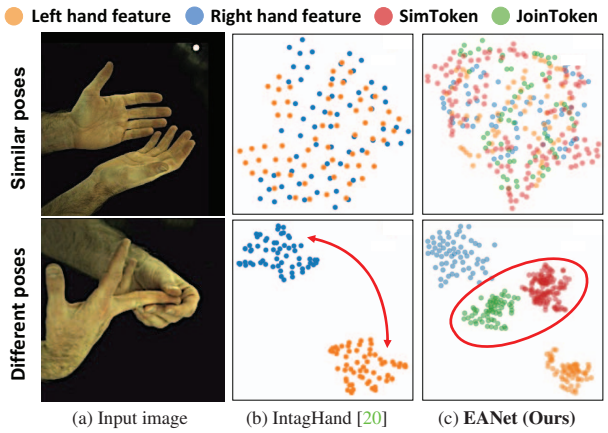
The main challenge for recovering the mesh of interacting hands lies in capturing the context of their interaction, which differs from recovering a single hand mesh. Recent works [12, 20, 8], inspired by the success of the attention mechanism [35, 9], use Transformer [35] to capture the correlation between the two hands. There have been two main approaches. Figure 1a shows the first approach [12]. It does not separate features, extracted from a backbone [14], to each hand. Instead, it directly uses features from the backbone as input tokens of a Transformer. Figure 1b shows the second approach [20, 8]. It separates features from a backbone to left and right hand-dedicated features. Then, it uses the separated features as input tokens of a Transformer.



(a) Keypoint Trans [12] **Separate L/R feat.: ✗ No distant token: ✓**
(b) IntagHand [20] **Separate L/R feat.: ✓ No distant token: ✗**
(c) **EABlock (Ours) Separate L/R feat.: ✓ No distant token: ✓**

Figure 1: **Comparison between previous Transformer blocks and our EABlock.** Instead of directly using feature from backbone or left and right hand features as input tokens, our FuseFormer in EABlock uses SimToken and JoinToken as input tokens.

Separating the feature from the backbone can make the network robust to the similarity between two hands. In addition, left and right hand-dedicated branches could focus only on one type of hand instead of two hands, which could relieve burden of modules [27, 20]. However, it can cause a *distant token problem*. Keypoint Transformer [12] in Figure 1a does not separate features, hence it does not suffer from the distant token problem; however, it cannot enjoy the benefit of the separation. In contrast, IntagHand [20] in Figure 1b suffers from the distant token problem as it separates features, while enjoying the benefit of the separation. The



(a) Input image (b) IntagHand [20] (c) **EANet (Ours)**

Figure 2: **Comparison of token distribution with t-SNE [34].** t-SNEs are drawn with input tokens, when given a single image of (a). (b) Previous methods [20] use left and right hand features as input tokens, which suffer from the distant token problem when poses of two hands are different. (c) We use SimToken and JoinToken as input tokens, suffering much less from the problem.

distant token problem arises due to the heterogeneous nature of input tokens projected to two separate spaces, causing the Transformer to struggle in capturing correlations between two hands. Figure 2b demonstrates the existence of the distant token problem between two hand features. Interestingly, we observed that the degree of heterogeneity between the two hand features becomes severe when the poses of the two hands are significantly different. The interaction between the two hands can still be important even when the poses of the hands are very different; however, previous works fail to capture such interaction effectively because of the heterogeneity of the two hand features. As a result, this leads to sub-optimal performance. Such distant token problem has also been addressed in the recent multi-modal learning community [28, 17, 21, 16]. While the distant token problem of two hands may not be as severe as in the case of multi-modality, it is still a challenge that needs to be addressed.

To address the distant token problem, we newly introduce two tokens: SimToken and JoinToken, which lie in homogeneous spaces, as shown in Figure 2c, containing two hand information. Instead of directly passing the separated left and right hand features to Transformers like previous works [20, 8], we convert the left hand and right hand features to new tokens and pass the converted ones to Transformers. The SimToken is obtained through **sim**ilarity-based operation using a self-attention (SA) Transformer [35]. The JoinToken is obtained by forwarding a concatenation of two input features to a fully connected layer; therefore, JoinToken models **join**t distribution of two input features without any similarity-based operations. As the new tokens are from both left and right hand features, they lie in homogeneous spaces. We design the new tokens to be complementary. For example, when two hands have very different poses, the distant token problem (the left and right hand features from bottom row of Figure 2c) limits the ability of SA Transformer [35] to capture their interaction. This can result in making SimToken less effective. On the other hand, JoinToken can still capture useful interaction information in such cases as it does not rely on similarity-based operations (*e.g.*, dot products in Transformer). Conversely, when two hands have similar poses, SimToken's SA mechanism enables it to capture highly useful interaction information that complements JoinToken. In this way, the new two tokens allow our system to 1) be robust to similarity between two hands by separating left and right hand features and 2) avoid the distant token problem and compute the correlation between input tokens properly.

We utilize the newly introduced two tokens in EANet, extract-and-adaptation network, where EABlock forms a main component of it. Figure 1c briefly shows EABlock, driven by our Transformer-based module, FuseFormer. FuseFormer fuses two input features by generating the two newly introduced tokens, SimToken and JoinToken, and processing them with a cross-attention (CA) Transformer.

Driven by the FuseFormer, the EABlock consists of extract and adaptation stages. In the extract stage, our EABlock first *extracts* an interaction feature from two hand features with a FuseFormer. In the adaptation stage, we *adapt* the extracted interaction feature to each hand with two additional FuseFormers for each hand. For the left hand adaptation as an example, we pass the extracted interaction feature and left hand feature to FuseFormer for fusing them. As the interaction feature mostly contains information about both hands, it may not be optimal for separately recovering the 3D hand mesh of each hand. Therefore, we fuse the interaction feature and left hand feature to achieve two objectives: 1) preserving the interaction information between the two hands and 2) obtaining more left hand-specific information. We follow a similar process for the right hand adaptation, passing the extracted interaction and right hand features through another FuseFormer.

With extensive experiments and qualitative analysis, we demonstrate the effectiveness of our framework. In the challenging InterHand2.6M [27] and HIC [33] datasets, we show that our EANet outperforms the previous state-of-the-art methods by a significant margin. Our contributions are summarized as follows:

- We propose EANet, extract-and-adaptation network, for 3D interacting hand mesh recovery. With the help of main block, EABlock, our system effectively captures the interaction between two hands even when poses of two hands are very different.

- We design FuseFormer, a core component of our EABlock. FuseFormer fuses input features by generating two complementary types of tokens, SimToken and JoinToken.

- Extensive experiments and qualitative analysis show that the proposed EANet outperforms previous state-of-the-art methods on 3D hand mesh benchmarks.

2. Related works

3D interacting hand mesh recovery. Several works [41, 40, 2, 27, 39, 12, 20] have been proposed to address 3D hand mesh recovery from monocular RGB image. InterHand2.6M dataset [27, 26] is the first dataset to show a potential for 3D hand mesh recovery in interacting hands by proposing the first large-scale high-resolution real RGB-based 3D hand pose dataset. Following the release of the dataset [27], Zhang *et al*. [39] proposed a method that estimates 3D hand meshes based on initially estimated 2.5D heatmap of interacting two hand joints. The initially estimated 3D hand meshes are then jointly refined in a cascaded manner to take advantage of multi-scale features extracted from pre-trained ResNet-50 [14] backbone. As the Transformer [35]-based mechanisms have emerged for various tasks in computer vision [9, 13], several methods [20, 12] have been proposed to effectively exploit Transformer [35] in 3D interacting hand mesh recovery. Keypoint Transformer [12] introduced a SA Transformer [35]-based module that performs global context-aware feature encoding for keypoints features. Stepping forward, IntagHand [20] proposed a CA Transformer [35]-based module to inherently model two hand correlation between two interacting hands.

Despite promising results, previous Transformer-based methods [20, 8] suffer from the distant token problem as they directly utilize two hand features as input tokens when poses of two hands are different. In contrast, our EANet is designed to address such distant token problem by using our two novel tokens as input tokens: SimToken and JoinToken. The two tokens are prepared and processed in our FuseFormer. By employing multiple FuseFormers in our main block, EABlock, the proposed EANet effectively *extracts* interaction feature and *adapts* it to each hand.

Tokenization in Transformer. Transformer [35]-based methods have become the de-facto standards in multiple tasks for both natural language processing [7, 3] and vision [9, 4, 13]. The rise of Transformer [35] has largely been accounted for its general purpose inductive bias, allowing flexibility [17, 1, 24] to handle input data while making minimal assumptions [17]. However, to compensate for such flexibility, many works have demonstrated task-specific add-ons such as positional embedding [31, 16, 22] or tokenization [9, 11, 1, 37, 24]. Especially, tokenization [9, 1, 37, 24] has been widely developed in vision as tokens in Transformer do not directly correspond to any single structure in an image. As demonstrated in vision, proper tokenization is essential for Transformer to make its full potential in corresponding tasks [38, 36]. In our work, we introduce two novel tokens named SimToken and JoinToken. The two novel tokens allow Transformer to overcome distant token problem of two input features by effectively fusing the input features.

3. Extract-and-adaptation network (EANet)

Figure 3 provides an overall architecture of our EANet. The proposed EANet consists of backbone, EABlock, joint feature extractor, self-joint Transformer, and regressor.

3.1. Backbone

Given an input hand image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, a backbone extracts hand features for both left hand $\mathbf{F}_{\mathrm{L}}$ and right hand $\mathbf{F}_{\mathrm{R}}$. $H = 256$ and $W = 256$ represent height and width of the input image, respectively. We first feed a hand image $\mathbf{I}$ to ResNet-50 [14], pre-trained on ImageNet [6], to extract an image feature $\mathbf{F} \in \mathbb{R}^{h\times w\times C}$, where $h = H/32$ and $w = W/32$ denote height and width of $\mathbf{F}$ respectively, and $C = 2048$ denotes the channel dimension of $\mathbf{F}$. Then, the image feature $\mathbf{F}$ is passed to two separate 1×1 convolutional layers to obtain left hand feature $\mathbf{F}_{\mathrm{L}}$ and right hand feature $\mathbf{F}_{\mathrm{R}}$. The two features have the same dimension of $\mathbb{R}^{h\times w\times c}$ where $c = C/4$.

3.2. Extract-and-adaptation block (EABlock)

Figure 4 shows detailed pipeline of our EABlock. EABlock consists of two stages (*i.e*., extract and adaptation), and FuseFormers are used in each stage. FuseFormer is our basic block, which fuses two types of input features with SimToken and JoinToken. FuseFormer consists of a SA Transformer, a fully connected layer, and a CA Transformer.

Extract: Interaction feature extract. Figure 5 shows the overall pipeline of FuseFormer in the extract stage. The FuseFormer in the extract stage *extracts* an interaction feature $\mathbf{F}_{\text{inter}}$ by fusing two hand features ($\mathbf{F}_{\mathrm{L}}$ and $\mathbf{F}_{\mathrm{R}}$). Different from the previous Transformer-based methods [12, 20] that utilize $\mathbf{F}_{\mathrm{L}}$ and $\mathbf{F}_{\mathrm{R}}$ as input tokens, we use SimToken $\mathbf{t}_{\mathrm{S}}$ and JoinToken $\mathbf{t}_{\mathrm{J}}$ as input tokens to address the distant token problem.

SimToken $\mathbf{t}_{\mathrm{S}}$ is obtained by passing the two hand features ($\mathbf{F}_{\mathrm{L}}$ and $\mathbf{F}_{\mathrm{R}}$) to a SA Transformer [35]. To this end, we first reshape $\mathbf{F}_{\mathrm{L}}$ and $\mathbf{F}_{\mathrm{R}}$ from $\mathbb{R}^{h\times w\times c}$ to $\mathbb{R}^{hw\times c}$. Then, we concatenate the reshaped $\mathbf{F}_{\mathrm{L}}$ and $\mathbf{F}_{\mathrm{R}}$ along with a class token $\mathbf{t}_{\text{cls}} \in \mathbb{R}^{1\times c}$ [9]. The class token is implemented as a learnable one-dimensional embedding vector to learn a general two hand information [9]. We denote the concatenated token by $\mathbf{t} \in \mathbb{R}^{l\times c}$, where $l = hw + hw + 1$. Then, we extract query $\mathbf{q}_{\mathrm{SA}}$, key $\mathbf{k}_{\mathrm{SA}}$, and value $\mathbf{v}_{\mathrm{SA}}$ from the concate-

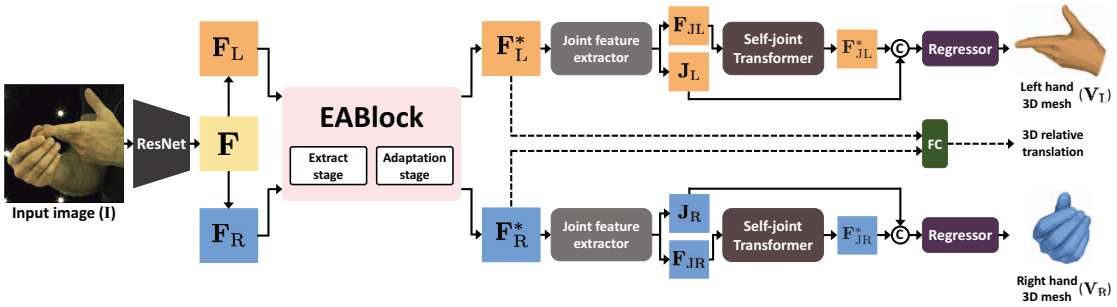


Figure 3: **The overall architecture of extract-and-adaptation network (EANet).** Our EANet *extracts* an interaction feature and *adapts* it to each hand. Then, from the adapted features ($\mathbf{F}^*_{\mathrm{L}}$ or $\mathbf{F}^*_{\mathrm{R}}$), joint feature extractors extract joint features of each hand ($\mathbf{F}_{\mathrm{JL}}$ or $\mathbf{F}_{\mathrm{JR}}$), guided by corresponding predicted 2.5D joint coordinates ($\mathbf{J}_{\mathrm{L}}$ or $\mathbf{J}_{\mathrm{R}}$). Finally, regressor produces MANO parameters, which are forwarded to MANO layers for reconstructing 3D hand meshes ($\mathbf{V}_{\mathrm{L}}$ or $\mathbf{V}_{\mathrm{R}}$). © denotes concatenation.

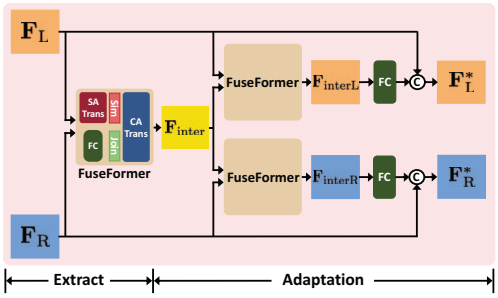


Figure 4: **The overall pipeline of EABlock.** A FuseFormer in the extract stage of EABlock first extracts an interaction feature $\mathbf{F}_{\mathrm{inter}}$ by using both hand features $\mathbf{F}_{\mathrm{L}}$ and $\mathbf{F}_{\mathrm{R}}$. Then, with an interaction feature $\mathbf{F}_{\mathrm{inter}}$ and one hand feature ($\mathbf{F}_{\mathrm{L}}$ or $\mathbf{F}_{\mathrm{R}}$), each FuseFormer in the adaptation stage of EABlock produces adapted interaction feature ($\mathbf{F}_{\mathrm{interL}}$ or $\mathbf{F}_{\mathrm{interR}}$). Afterwards, adapted interaction features are concatenated with corresponding hand features to produce adapted hand feature ($\mathbf{F}^*_{\mathrm{L}}$ or $\mathbf{F}^*_{\mathrm{R}}$). © denotes concatenation.

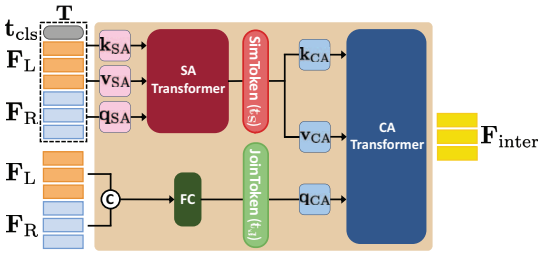


Figure 5: **The overall pipeline of FuseFormer in extract stage.** FuseFormer first prepares SimToken $\mathbf{t}_{\mathrm{S}}$ and JoinToken $\mathbf{t}_{\mathrm{J}}$ with SA Transformer and a fully connected layer, respectively. Then, a CA Transformer processes the two tokens. © denotes concatenation.

nated token $\mathbf{t}$ with separate linear layers. The dimensions of $\mathbf{q}_{\mathrm{SA}}$, $\mathbf{k}_{\mathrm{SA}}$, and $\mathbf{v}_{\mathrm{SA}}$ are all identical to that of $\mathbf{t}$. Analogous to the previous Transformers [35, 9], the SimToken $\mathbf{t}_{\mathrm{S}} \in \mathbb{R}^{l \times c}$ is produced as follows:

$$\mathrm{Attn}(\mathbf{q}_{\mathrm{SA}}, \mathbf{k}_{\mathrm{SA}}, \mathbf{v}_{\mathrm{SA}}) = \mathrm{softmax}(\frac{\mathbf{q}_{\mathrm{SA}}\mathbf{k}_{\mathrm{SA}}{}^{T}}{\sqrt{d_{\mathbf{k}_{\mathrm{SA}}}}})\mathbf{v}_{\mathrm{SA}}, \quad (1)$$

$$\mathbf{r} = \mathbf{t} + \mathrm{Attn}(\mathbf{q}_{\mathrm{SA}}, \mathbf{k}_{\mathrm{SA}}, \mathbf{v}_{\mathrm{SA}}), \quad (2)$$

$$\mathbf{t}_{\mathrm{S}} = \mathbf{r} + \mathrm{MLP}(\mathbf{r}), \quad (3)$$

where $d_{\mathbf{k}_{\mathrm{SA}}}$ denotes a channel dimension of $\mathbf{k}_{\mathrm{SA}}$. Consequently, $\mathbf{t}_{\mathrm{S}}$ is built upon a standard SA Transformer from two hand features.

JoinToken $\mathbf{t}_{\mathrm{J}} \in \mathbb{R}^{hw \times c}$ is obtained by passing two hand features ($\mathbf{F}_{\mathrm{L}}$ and $\mathbf{F}_{\mathrm{R}}$) to a fully connected layer. Before passing the two hand features, we concatenate and reshape two hands features ($\mathbf{F}_{\mathrm{L}}$ and $\mathbf{F}_{\mathrm{R}}$) from $\mathbb{R}^{h \times w \times 2c}$ to $\mathbb{R}^{hw \times 2c}$. Formally,

$$\mathbf{t}_{\mathrm{J}} = \mathrm{FC}(\psi(\mathbf{F}_{\mathrm{L}}, \mathbf{F}_{\mathrm{R}})), \quad (4)$$

where ψ represents a composition of concatenation and reshape functions. FC denotes a fully connected layer.

After preparing two tokens ($\mathbf{t}_{\mathrm{S}}$ and $\mathbf{t}_{\mathrm{J}}$), a CA [5] Transformer extracts an interaction feature $\mathbf{F}_{\mathrm{inter}} \in \mathbb{R}^{hw \times c}$ from the two tokens. For CA, we extract query from JoinToken $\mathbf{t}_{\mathrm{J}}$ and key-value pair from SimToken $\mathbf{t}_{\mathrm{S}}$ with separate linear layers. We denote query, key and value of the CA Transformer by $\mathbf{q}_{\mathrm{CA}} \in \mathbb{R}^{hw \times c}$, $\mathbf{k}_{\mathrm{CA}} \in \mathbb{R}^{l \times c}$, and $\mathbf{v}_{\mathrm{CA}} \in \mathbb{R}^{l \times c}$, respectively.

The reason for using JoinToken as a query is to address *mismatch between the query-key correlation and value*. Let us take an example where the inputs of the CA Transformer are obtained in an opposite way: query from SimToken and key-value from JoinToken. When poses of two hands are very different, SimToken fails to capture useful interaction between two hands; hence, first half and second half of SimToken mainly contain the left and right hand information, respectively, not both hand information. On the other hand, JoinToken contains information of both hands. Let us take a single element in query, which mostly contains left hand information, as an example. From this element's point of view, the query-key correlation is always formulated by how much all elements in JoinToken are similar to the left hand information. However, as value is from JoinToken,

the opposite hand information can be retrieved based on the left hand-driven similarity, which is undesirable. In the end, using SimToken as a query suffers from the mismatch between query-key correlation (driven by the left hand) and value (right hand information). In contrast, ours extracts query from JoinToken and key-value from SimToken. As elements in query contain both hand information, the query-key correlation is not fixed to a single type of hand (*e.g.*, left hand); hence, ours does not suffer from the mismatch.

In the end, the CA Transformer outputs an interaction feature $\mathbf{F}_{\text{inter}}$ following Equation 1, 2, and 3 with $\mathbf{q}_{\text{CA}}$, $\mathbf{k}_{\text{CA}}$, and $\mathbf{v}_{\text{CA}}$. Consequently, FuseFormer in the extract stage *extracts* interaction feature $\mathbf{F}_{\text{inter}}$ by fusing two hand features ($\mathbf{F}_{\text{L}}$ and $\mathbf{F}_{\text{R}}$). Formally,

$$\mathbf{F}_{\text{inter}} = \text{FuseFormer}(\mathbf{F}_{\text{L}}, \mathbf{F}_{\text{R}}; \mathbf{W}_{\text{extract}}), \tag{5}$$

where $\mathbf{W}_{\text{extract}}$ denotes learnable weights of the FuseFormer in the extract stage.

Adaptation: Interaction feature adaptation. In the adaptation stage, EABlock adapts the extracted interaction feature $\mathbf{F}_{\text{inter}}$ to each hand with two additional FuseFormers. While the interaction feature, obtained from the extract stage, is useful for understanding how two hands interact with each other, directly utilizing it for 3D hand mesh recovery of each hand might not be optimal. Therefore, we fuse the interaction feature and feature of each hand to achieve two goals: 1) preserving interaction information between two hands and 2) obtaining left and right hand-specific information. Taking the left hand adaptation as an example, we pass the left hand feature $\mathbf{F}_{\text{L}}$ and the interaction feature $\mathbf{F}_{\text{inter}}$ to a FuseFormer, which fuses them and outputs interaction feature adapted to the left hand $\mathbf{F}_{\text{interL}} \in \mathbb{R}^{hw \times c}$. The adaptation for the right hand is performed in the same manner by passing the right hand feature $\mathbf{F}_{\text{R}}$ and interaction feature $\mathbf{F}_{\text{inter}}$ to another FuseFormer, which fuses them and outputs interaction feature adapted to the right hand $\mathbf{F}_{\text{interR}} \in \mathbb{R}^{hw \times c}$. Formally,

$$\mathbf{F}_{\text{interL}} = \text{FuseFormer}(\mathbf{F}_{\text{L}}, \mathbf{F}_{\text{inter}}; \mathbf{W}_{\text{adaptL}}), \tag{6}$$

$$\mathbf{F}_{\text{interR}} = \text{FuseFormer}(\mathbf{F}_{\text{R}}, \mathbf{F}_{\text{inter}}; \mathbf{W}_{\text{adaptR}}), \tag{7}$$

where $\mathbf{W}_{\text{adaptL}}$ and $\mathbf{W}_{\text{adaptR}}$ denote learnable weights in FuseFormers for the left and right hand adaptations, respectively.

To reduce computational costs, for each hand, we apply a fully connected layer, which reduces the channel dimension from c to $c/4$, to the adapted interaction features ($\mathbf{F}_{\text{interL}}$ or $\mathbf{F}_{\text{interR}}$). Then, we reshape the output of the fully connected layer to $\mathbb{R}^{h \times w \times c/4}$. Finally, for each hand, we concatenate the adapted interaction feature ($\mathbf{F}_{\text{interL}}$ or $\mathbf{F}_{\text{interR}}$) and its corresponding hand feature ($\mathbf{F}_{\text{L}}$ or $\mathbf{F}_{\text{R}}$) along the channel dimension, which becomes final adapted feature of each hand, denoted by $\mathbf{F}^*_{\text{L}}$ and $\mathbf{F}^*_{\text{R}}$.

3.3. Joint feature extractor

For each hand, a joint feature extractor extracts 2.5D joint coordinates ($\mathbf{J}_{\text{L}}$ or $\mathbf{J}_{\text{R}}$) and joint features ($\mathbf{F}_{\text{JL}}$ or $\mathbf{F}_{\text{JR}}$) from the adapted hand features ($\mathbf{F}^*_{\text{L}}$ or $\mathbf{F}^*_{\text{R}}$). Motivated by Moon *et al*. [25], our joint feature extractor effectively extracts joint features with the guidance of estimated 2.5D joint coordinates.

2.5D joint coordinate estimation. x- and y-axis of 2.5D joint coordinates are in the 2D pixel space, while z-axis of them are in the root joint (*i.e.*, wrist)-relative depth space. To estimate 2.5D joint coordinates of each hand, we apply a 1×1 convolution layer to the adapted hand features ($\mathbf{F}^*_{\text{L}}$ or $\mathbf{F}^*_{\text{R}}$) to change their channel dimension to dJ. Then, we reshape the output to a 2.5D heatmap whose dimension is $\mathbb{R}^{h \times w \times d \times J}$. $d = 8$ denotes discretized depth size of the 2.5D heatmap, and J denotes the number of joints. Subsequently, 2.5D joint coordinates for left or right hand ($\mathbf{J}_{\text{L}}$ or $\mathbf{J}_{\text{R}}$) are obtained by applying the soft-argmax operation [32] to the corresponding 2.5D heatmap.

Joint feature extract. The left hand joint features $\mathbf{F}_{\text{JL}}$ are obtained through a grid sampling [15] on feature map $\mathbf{F}^*_{\text{L}}$ with (x, y) position from the estimated 2.5D joint coordinates $\mathbf{J}_{\text{L}}$. Likewise, the right hand joint features $\mathbf{F}_{\text{JR}}$ are obtained in a same manner with right hand feature $\mathbf{F}^*_{\text{R}}$ and the corresponding 2.5D joint coordinates $\mathbf{J}_{\text{R}}$. The joint features contain global contextual information of each hand joint, essential for 3D hand mesh recovery.

3.4. Self-joint Transformer (SJT)

Self-joint Transformer (SJT) is a standard SA Transformer. For each hand, it enhances joint features ($\mathbf{F}_{\text{JL}}$ or $\mathbf{F}_{\text{JR}}$) through the SA [35], which outputs ($\mathbf{F}^*_{\text{JL}}$ or $\mathbf{F}^*_{\text{JR}}$). Despite its simplicity, we observed that SJT is greatly helpful to enhance the joint features as it can implicitly consider kinematic structure of hand joints through the SA. Note that the goal of SJT is clearly different from that of FuseFormer: FuseFormer aims to fuse two types of input features, while the SJT aims to enhance a single type of input feature.

3.5. Final outputs

For each hand, the regressor produces 48-dimensional pose parameters (θ_{L} or θ_{R}) and 10-dimensional shape parameters (β_{L} or β_{R}) of MANO [30] from the enhanced hand joint features ($\mathbf{F}^*_{\text{JL}}$ or $\mathbf{F}^*_{\text{JR}}$). The pose parameters are obtained by first concatenating the enhanced joint features ($\mathbf{F}^*_{\text{JL}}$ or $\mathbf{F}^*_{\text{JR}}$) with 2.5D joint coordinates ($\mathbf{J}_{\text{L}}$ or $\mathbf{J}_{\text{R}}$). Then, we flattened the concatenated features each into a one-dimensional vector. The final two vectors are passed to two fully connected layers to predict MANO pose parameters. To obtain the shape parameters, we forward the adapted hand features ($\mathbf{F}^*_{\text{L}}$ or $\mathbf{F}^*_{\text{R}}$) to a fully connected layer after global average pooling [23]. In the end, the final 3D

Separate L/R feature	EABlock	MPJPE			MPVPE			MRRPE
		Single	Two	All	Single	Two	All	
✗	✗	14.90	15.73	15.32	11.84	14.05	12.80	22.76
✓	✗	12.74	14.33	13.53	12.53	11.95	12.28	23.79
✓	✓	**9.62**	**11.54**	**10.58**	**7.86**	**10.78**	**9.13**	**18.82**

Table 1: **Comparison of models with various architectures in Figure 1.**

Block	MPJPE			MPVPE			MRRPE
	Single	Two	All	Single	Two	All	
CA Transformer	12.74	14.33	13.53	12.53	11.95	12.28	23.79
SA Transformer	11.44	13.49	12.47	9.51	12.37	10.75	25.35
Ours (w/o adaptation)	10.66	11.81	11.25	8.37	11.24	9.61	21.00
Ours (full)	**9.62**	**11.54**	**10.58**	**7.86**	**10.78**	**9.13**	**18.82**

Table 2: **Comparison of models with various Transformer blocks for EABlock.**

hand meshes, denoted by $\mathbf{V}_{\mathrm{L}}$ and $\mathbf{V}_{\mathrm{R}}$, are obtained by forwarding the MANO parameters to MANO layers. In addition, the 3D relative translation between two hands is obtained from the adapted hand features ($\mathbf{F}_{\mathrm{L}}^{*}$ and $\mathbf{F}_{\mathrm{R}}^{*}$). To this end, we concatenate them and perform global average pooling. Finally, a fully connected layer is used to output the 3D relative translation between two hands. To train our model, we minimize loss functions, defined as a weighted sum of L1 distances between estimated and ground truth θ, β, $\mathbf{J}$, $\mathbf{V}$, and 3D relative translation.

4. Experiments

4.1. Implementation details

All implementations are done with PyTorch [29] under Adam optimizer [19] and in batch size of 32 per GPU (trained with four RTX 2080 Ti GPUs). For training our model on InterHand2.6M [27] dataset, we trained our model for 30 epochs, with learning rate annealing at 10th and 15th epochs from the initial learning rate 1×10^{-4}.

4.2. Datasets and evaluation metrics

InterHand2.6M dataset. We trained and evaluated EANet on InterHand2.6M dataset. For the comparison with the previous approaches, we used both single and interacting hand (SH+IH) images of the H+M split. For ablation studies, we used SH+IH images of the H split.

HIC dataset. To show the generalizability of our proposed EANet, we further showed results on HIC [33] dataset. Different from InterHand2.6M [27] dataset, which has homogeneous background and lighting sources, HIC dataset includes natural lighting with more diverse backgrounds. Note that the HIC dataset was only used for evaluation.

Evaluation metrics. Following the prior arts [27, 12], we reported mean per joint position error (MPJPE), mean per vertex position error (MPVPE), and mean relative-root position error (MRRPE). All metrics are in a millimeter scale.

(a) Input image (b) CA Transformer (c) SA Transformer (d) Ours (w/o adapt.) (e) **Ours (full)**

Figure 6: **Visual comparison of models with various Transformer blocks for EABlock.**

4.3. Ablation studies

Overall design of EANet. Table 1 justifies our approach, which 1) separates left and right hand features and 2) uses EABlock to address the distant token problem, as depicted in Figure 1c. For the first row, we did not separate the feature from the backbone to left and right hand features. Instead, the feature from the backbone was directly passed to SA Transformer following Keypoint Transformer [12]. Then, a single joint feature extractor and self-joint Transformer were used. Two regressors for each hand output final 3D hand meshes. For the second row, we replaced EABlock with CA Transformer following IntagHand [20] while keeping the separation. Without the separation (Figure 1a and the first row of the table), model's performance drops significantly as the separation is helpful to make the network robust to the similarity between left and right hands. Also, left and right hand-dedicated branches could focus only on one type of hands instead of two hands, which could relieve the burden of modules [27, 20]. Compared to the second row of the table (Figure 1b), ours addresses the distant token problem by introducing the EABlock, which results in performance improvements.

EABlock vs. previous Transformer blocks. Table 2 and Figure 6 demonstrate the superiority of the EABlock over previous Transformer blocks. For the first and second rows of Table 2, we replaced our EABlock with standard Transformers with a similar number of parameters as EABlock, each with corresponding SA and CA modules. Following the previous Transformer-based methods [20, 12], we used two hand features as the input tokens. The SA Transformer (the second row) extracts query, key, and value from both hands, and CA Transformer (the first row) extracts query from one hand and key-value pair from the other hand. As shown, SA Transformer shows better results than CA Transformer, especially in the single hand cases. This is because the CA mechanism forcefully injects irrelevant information from non-existing counterpart hand.

Meanwhile, the third row shows that our EABlock outperforms the two Transformer baselines only with the extract stage (*i.e.*, without the adaptation stage) in EABlock. Especially, the gain in two hands cases (+1.68) is larger than that of single hand cases (+0.78) in terms of MPJPE. This demonstrates the benefits of our two novel tokens, which are proposed to relieve the distant token problem. With the adaptation stage, our EABlock improves its performance in

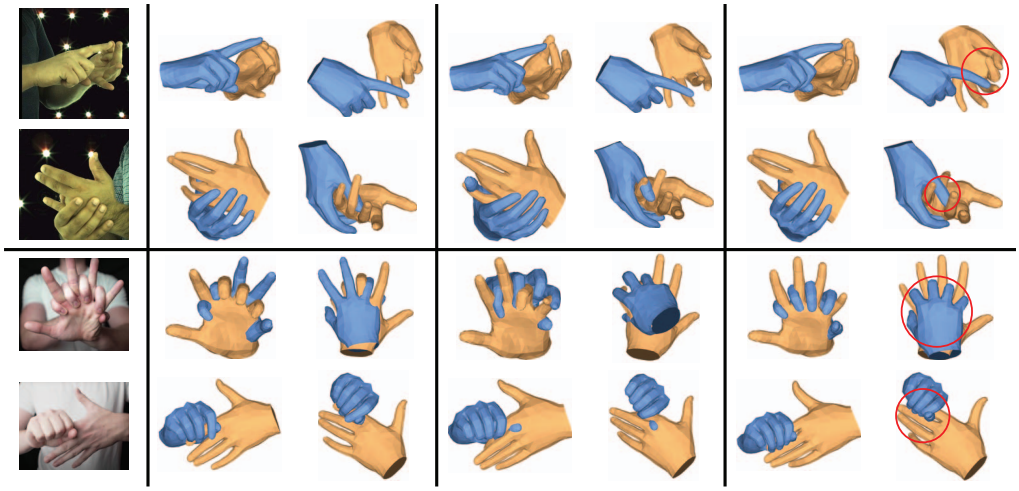


Figure 7: **Visual comparison with state-of-the-art methods on InterHand2.6M [27] (top) and in-the-wild (bottom).** The red circles highlight regions where our EANet is correct, while others are wrong. We crawl in-the-wild images from web.

$\mathbf{q}_{CA}$	$\mathbf{k}_{CA}, \mathbf{v}_{CA}$	MPJPE			MPVPE			MRRPE
		Single	Two	All	Single	Two	All	
-	-	13.19	14.17	13.88	9.68	13.61	11.95	25.95
$\mathbf{t}_J$	$\mathbf{F}_L, \mathbf{F}_R$	10.66	12.60	11.63	8.73	11.65	10.00	21.88
$\mathbf{t}_J$	$\mathbf{t}_J$	12.02	13.42	12.71	8.79	11.63	10.02	22.67
$\mathbf{t}_S$	$\mathbf{t}_S$	10.44	12.36	11.40	8.53	11.33	9.75	21.70
$\mathbf{t}_S$	$\mathbf{t}_J$	10.48	12.40	11.44	8.58	11.37	9.79	21.52
$\mathbf{t}_J$	$\mathbf{t}_S$	**9.62**	**11.54**	**10.58**	**7.86**	**10.78**	**9.13**	**18.82**

Table 3: **Comparisons of models with various inputs for CA Transformer in FuseFormer.** The first row uses $\mathbf{t}_J$ as the interaction feature without using the CA Transformer.

EABlock	SJT	MPJPE			MPVPE			MRRPE
		Single	Two	All	Single	Two	All	
✗	✗	11.22	13.18	12.20	10.08	12.87	11.28	23.04
✗	✓	11.05	13.04	12.04	9.07	11.92	10.28	22.80
✓	✗	10.12	11.92	11.02	8.61	11.42	9.83	21.11
✓	✓	**9.62**	**11.54**	**10.58**	**7.86**	**10.78**	**9.13**	**18.82**

Table 4: **Comparison of models with various combinations of EABlock and SJT.**

Methods	# of params (M)	MPVPE			MRRPE
		Single	Two	All	
Zhang *et al*. [39]	143.37	-	13.95	-	-
Keypoint Transformer [12]	117.05	10.16	14.36	11.94	30.87
IntagHand [20]	39.04	-	9.03	-	-
EANet (Light. Ours)	**33.90**	5.71	7.66	6.53	34.29
EANet (Ours)	106.82	**4.81**	**6.34**	**5.45**	**28.54**

Table 5: **Quantitative comparison with state-of-the-art methods on InterHand2.6M [27] dataset.**

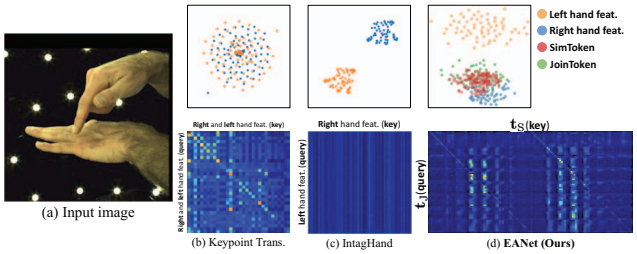
Methods	MPVPE			MRRPE
	Single	Two	All	
Zhang *et al*. [39]	-	42.08	-	-
Keypoint Transformer [12]	60.19	51.21	54.71	190.77
IntagHand [20]	-	45.74	-	-
EANet (Light. Ours)	43.90	38.47	40.59	83.44
EANet (Ours)	**32.82**	**34.43**	**33.80**	**81.11**

Table 6: **Quantitative comparison with state-of-the-art methods on HIC [33] dataset.**

all metrics, shown in the fourth row. In particular, the adaptation has more gain in single hand cases compared to two hand cases, both for MPJPE and MPVPE. This justifies our adaptation stage, designed to adapt the interaction feature to each hand and filter irrelevant information from the interaction feature.

Inputs of CA Transformer in FuseFormer. Table 3 justifies our strategy to use JoinToken $\mathbf{t}_J$ and SimToken $\mathbf{t}_S$ as query and key-value pair of the CA Transformer in FuseFormer, respectively. The comparison between the fifth row and our setting (the last row) shows that using JoinToken as a query produces lower 3D errors compared to using SimToken as a query. This shows our setting effectively addresses the *mismatch between query-key correlation and value*, described in Section 3.2. In addition, our setting performs better than using a single type of token (the third and fourth rows), which indicates that our two tokens are complementary. Ours also produces lower 3D errors compared to the second row that uses two hand features ($\mathbf{F}_L$ and $\mathbf{F}_R$) as a key-value pair. This proves the benefit of using SimToken compared to the two hand features. Lastly, ours produces far better results than the first row that uses JoinToken as the interaction feature without using the CA Transformer in FuseFormer. This demonstrates that solely using JoinToken is insufficient, and the CA Transformer is necessary to fuse two types of input features of FuseFormer.

Combinations of EABlock and SJT. Table 4 shows that our SJT greatly harmonizes with the proposed EABlock. Solely employing the SJT without EABlock, in the second row, shows a minor improvement. Adopting EABlock alone, in the third row, shows considerable improvement compared to the baseline. Most importantly, jointly adopting both EABlock and SJT further consistently improves the performance in every metric.



(a) Input image (b) Keypoint Trans. (c) IntagHand (d) **EANet (Ours)**

Figure 8: **Comparison of t-SNEs and attention maps.** Ours is from CA Transformer in extract stage of EABlock.

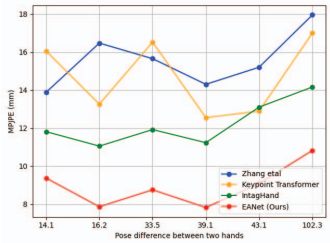


Figure 9: **Comparison on InterHand2.6M [27] sequences with various pose differences.** x-axis represents pose difference between two hands. y-axis represents errors of the methods on each sequence with pose difference of x-axis.

4.4. Comparisons with state-of-the-art methods

Comparison on InterHand2.6M and HIC. Table 5 and 6 show that our EANet achieves the highest performance on Interhand2.6M [27] and HIC [33] datasets, respectively. Following the previous works [20, 39], we measured MPVPEs after scale alignment on meshes with GTs. Due to the absence of MPVPE in the manuscript of Keypoint Transformer, the numbers were obtained from their officially released codes and pre-trained weights. In Table 5, we compare the performance and the number of parameters with previous 3D interacting hand mesh recovering methods [27, 25, 10, 12]. Our EANet has fewer parameters than Keypoint Transformer [12] and Zhang *et al*. [39], while significantly outperforming them. Moreover, to ensure a fair comparison with IntagHand [20], which has fewer parameters than ours, we introduce a lighter version of EANet, EANet (Light). We reduced the channel dimension to $c = C/8$ instead of $c = C/4$ for EANet (Light). Even with fewer parameters, EANet (Light) shows better MPVPE than IntagHand [20], further demonstrating the efficacy of our proposed approach. Furthermore, in Table 6, our EANet achieves significantly better performance on HIC dataset than other methods, indicating its strong generalizability.

In addition, Figure 7 shows the visual comparisons with previous state-of-the-art methods [20, 12] on InterHand2.6M [27] and in-the-wild interacting hand images. The results showcase our EANet's superior performance in precise hand pose and interaction estimation.

Attention map comparison. Figure 8 compares attention maps and corresponding t-SNEs. In Figure 8b, Keypoint Transformer [12] does not suffer from the distant token problem, as it does not separate feature from the backbone to left and right hand features. However, without the separation, networks suffer from sub-optimal performance, as shown in Table 1. Also, its attention map shows the dominance of strong self-correlations with weak correlations from non-diagonal areas. In Figure 8c, IntagHand suffers from the distant token problem. Hence, it fails to generate proper correlations between two hands with different poses, resulting in predominantly low correlations between all queries and keys. In contrast, Figure 8d shows that our SimToken and JoinToken do not suffer from the distant token problem with balanced high correlations in both the left- and right-half of the attention map. As the each half of the attention map was computed between each query and each hand portion of SimToken, the high correlations in both halves indicate that ours effectively utilizes both left and right hand information for each query.

Robustness to asymmetric poses of two hands. Figure 9 demonstrates that our EANet exhibits significantly greater robustness to asymmetric poses of the hands compared to previous state-of-the-art methods [39, 12, 20]. For this analysis, we selected 6 sequences from the InterHand2.6M [27], which have diverse pose similarities between the two hands. The x-axis of the figure represents the average pose difference between the two hands in each selected sequence, with larger values indicating bigger pose differences. Notably, our EANet produces nearly consistent MPJPE values across all pose differences, even when the pose difference increases about 7 times at the rightmost x value compared to the leftmost one. Moreover, our method produces significantly lower MPJPE values than previous methods for all sequences, including those with the biggest pose differences. We clarify the way to calculate the pose difference of each sequence in the supplementary material.

5. Conclusion

We present EANet, extract-and-adaptation network for 3D interacting hand mesh recovery. With the help of the main block, EABlock, EANet effectively learns interaction between two hands by solving distant token problem. The EABlock is mainly driven by our novel Transformer-based block, FuseFormer, which fuses two types of input features by using SimToken and JoinToken. By extracting interaction from two hand features and then adapting the interaction towards each hand in EABlock, our EANet achieves the state-of-the-art performance in recovering 3D interacting hand mesh on challenging scenarios.

Acknowledgments. This work was supported in part by the IITP grants [No.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University), No. 2021-0-02068, and No.2023-0-00156] and the NRF grant [No. 2021M3A9E4080782] funded by the Korea government (MSIT).

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *ICCV*, 2021. 3

[2] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 3D hand shape and pose from images in the wild. In *CVPR*, 2019. 3

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 3

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 3

[5] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. CrossViT: Cross-attention multi-scale Vision Transformer for image classification. In *ICCV*, 2021. 4

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 3

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 3

[8] Xinhan Di and Pengqian Yu. LWA-HAND: Lightweight attention hand for interacting hand reconstruction. In *ECCVW*, 2023. 1, 2, 3

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 3, 4

[10] Zicong Fan, Adrian Spurr, Muhammed Kocabas, Siyu Tang, Michael J Black, and Otmar Hilliges. Learning to disambiguate strongly interacting hands via probabilistic per-pixel part segmentation. In *3DV*, 2021. 1, 8

[11] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *NeurIPS*, 2021. 3

[12] 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. 1, 3, 6, 7, 8

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 3

[15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NeurIPS*, 2015. 5

[16] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver IO: A general architecture for structured inputs & outputs. In *ICLR*, 2022. 2, 3

[17] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021. 2, 3

[18] Dong Uk Kim, Kwang In Kim, and Seungryul Baek. End-to-end detection and pose estimation of two interacting hands. In *ICCV*, 2021. 1

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 6

[20] 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. 1, 2, 3, 6, 7, 8

[21] Tao Liang, Guosheng Lin, Lei Feng, Yan Zhang, and Fengmao Lv. Attention is not enough: Mitigating the distribution discrepancy in asynchronous multimodal sequence fusion. In *ICCV*, 2021. 2

[22] Tatiana Likhomanenko, Qiantong Xu, Gabriel Synnaeve, Ronan Collobert, and Alex Rogozhnikov. CAPE: Encoding relative positions with continuous augmented positional embeddings. In *NeurIPS*, 2021. 3

[23] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 5

[24] Rui Liu, Hanming Deng, Yangyi Huang, Xiaoyu Shi, Lewei Lu, Wenxiu Sun, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. FuseFormer: Fusing fine-grained information in transformers for video inpainting. In *ICCV*, 2021. 3

[25] Gyeongsik Moon, Hongsuk Choi, and Kyoung Mu Lee. Accurate 3D hand pose estimation for whole-body 3D human mesh estimation. In *CVPRW*, 2022. 5, 8

[26] Gyeongsik Moon, Hongsuk Choi, and Kyoung Mu Lee. NeuralAnnot: Neural annotator for 3D human mesh training sets. In *CVPRW*, 2022. 3

[27] 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. 1, 2, 3, 6, 7, 8

[28] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. In *NeurIPS*, 2021. 2

[29] 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. 6

[30] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. In *SIGGRAPH Asia*, 2017. 5

[31] Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In *NeurIPS*, 2019. 3

[32] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *ECCV*, 2018. 5

[33] 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. 2, 6, 7, 8

[34] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *JMLR*, 2008. 2

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 3, 4, 5

[36] Hanqi Yan, Lin Gui, Wenjie Li, and Yulan He. Addressing token uniformity in Transformers via singular value transformation. In *UAI*, 2022. 3

[37] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-Token ViT: Training Vision Transformers from scratch on ImageNet. In *ICCV*, 2021. 3

[38] Wang Zeng, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Not all tokens are equal: Human-centric visual analysis via token clustering Transformer. In *CVPR*, 2022. 3

[39] 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. 1, 3, 7, 8

[40] Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. Monocular real-time hand shape and motion capture using multi-modal data. In *CVPR*, 2020. 3

[41] Christian Zimmermann and Thomas Brox. Learning to estimate 3D hand pose from single RGB images. In *ICCV*, 2017. 3