NeuralAnnot: Neural Annotator for 3D Human Mesh Training Sets

Gyeongsik Moon[1] Hongsuk Choi[1] Kyoung Mu Lee[1,2]

[1]Dept. of ECE & ASRI, [2]IPAI, Seoul National University, Korea

{mks0601,redarknight,kyoungmu}@snu.ac.kr

Abstract

Most 3D human mesh regressors are fully supervised with 3D pseudo-GT human model parameters and weakly supervised with GT 2D/3D joint coordinates as the 3D pseudo-GTs bring great performance gain. The 3D pseudo-GTs are obtained by annotators, systems that iteratively fit 3D human model parameters to GT 2D/3D joint coordinates of training sets in the pre-processing stage of the regressors. The fitted 3D parameters at the last fitting iteration become the 3D pseudo-GTs, used to fully supervise the regressors. Optimization-based annotators, such as SMPLify-X, have been widely used to obtain the 3D pseudo-GTs. However, they often produce wrong 3D pseudo-GTs as they fit the 3D parameters to GT of each sample independently. To overcome the limitation, we present NeuralAnnot, a neural network-based annotator. The main idea of NeuralAnnot is to employ a neural network-based regressor and dedicate it for the annotation. Assuming no 3D pseudo-GTs are available, NeuralAnnot is weakly supervised with GT 2D/3D joint coordinates of training sets. The testing results on the same training sets become 3D pseudo-GTs, used to fully supervise the regressors. We show that 3D pseudo-GTs of NeuralAnnot are highly beneficial to train the regressors. We made our 3D pseudo-GTs publicly available.

1. Introduction

3D human mesh estimation aims to localize human mesh vertices in the 3D space. Two types of datasets have been used by most 3D human mesh regressors [2, 9, 11, 19, 20, 26]. The first dataset type is motion capture (MoCap) datasets [4, 6, 17, 21, 29], captured with well-calibrated multiple cameras in highly restricted environment (*e.g.*, MoCap studio). Groundtruth (GT) 3D joint coordinates can be obtained owing to the special equipment; however, the restricted environment makes captured images have monotonous appearances. The second dataset type is in-the-wild datasets [5, 15]. They have images with diverse appearances as they are captured in our daily life without the special equipment. However, only GT 2D joint coordinates can be obtained by manual human labor, and 3D GTs are not obtainable because of depth and scale ambiguity. The GT 2D/3D joint coordinates are used to weakly supervise the 3D human mesh regressors.

Methods	Supervision targets	Where to test
One-stage annotator ([24],Ours)	**GT 2D/3D joint coords.**	**Training set**
Two-stage annotator [7, 11]	GT 2D/3D joint coords. + initial 3D pseudo-GTs	**Training set**
Regressor [9, 12, 20]	GT 2D/3D joint coords. + 3D pseudo-GTs	Test set

Table 1. Comparison of one-stage annotator (ours), two-stage annotator, and regressor.

In addition to the weak supervisions with GT 2D/3D joint coordinates, the regressors are fully supervised with 3D pseudo-GT human model parameters. SMPL body model [16], MANO hand model [25], FLAME face model [14], and SMPL-X [24] whole-body model are widely used 3D human models. Their parameters (*i.e.*, 3D pseudo-GTs) include 3D joint rotations and latent codes of body shapes, hand shapes, or facial expressions. "pseudo" represents that the parameters are obtained by being fit to GT 2D/3D joint coordinates; therefore, they are not true GTs. Although the 3D pseudo-GTs can have fitting errors, they complement the GT 2D/3D joint coordinates in two ways. First, 3D pseudo-GTs are full supervision targets, while GT 2D/3D joint coordinates are weak supervision targets, for the 3D human mesh regressors. This is because 3D pseudo-GTs of the input human represent a single 3D human mesh without any ambiguity. On the other hand, 2D/3D joint coordinates of the input human can represent multiple 3D human meshes. For example, 3D joint coordinates cannot determine a single roll-axis rotation of each joint. Furthermore, 2D joint coordinates suffer from severe depth ambiguity; therefore, multiple 3D meshes can be projected to the 2D joint coordinates. Second, 3D pseudo-GTs can provide 3D annotations for in-the-wild datasets. As in-the-wild datasets contain only GT 2D joint coordinates without 3Ds, 3D pseudo-GTs of in-the-wild datasets can resolve the most major bottleneck of 3D human mesh estimation: lack of 3D data in the wild. As the two points make 3D pseudo-GTs bring large performance improvements, most 3D human mesh estimation regressors [2, 9, 11, 20, 26] are fully supervised with 3D pseudo-GTs and weakly supervised with GT 2D/3D joint coordinates.

The 3D pseudo-GTs are obtained by annotators, systems that iteratively fit the 3D human model parameters to GT 2D/3D joint coordinates of training sets in the pre-processing stage of the regressors. The fitted 3D parameters at the last fitting iteration become the 3D pseudo-GTs. After fully supervised with the 3D pseudo-GTs and weakly supervised with GT 2D/3D joint coordinates, the regressors are tested on unseen samples in testing sets. Table 1 shows comparison of one-stage annotator, two-stage annotator, and regressor. The one-stage annotators, to which SMPLify-X [24] belongs, output 3D pseudo-GTs without initial 3D pseudo-GTs. SMPLify-X fits 3D human model parameters of T-pose to GT 2D joint coordinates of each sample. Let us denote the annotation procedure of the one-stage annotator as f_1. On the other hand, the two-stage annotators, to which SPIN [11] and EFT [7] belong, first obtain initial 3D pseudo-GTs from the one-stage annotators in the pre-processing stage of the two-stage annotators. This is the first stage of the two-stage annotator, which is exactly same with the annotation procedure of the one-stage annotator: f_1. Then, in the second stage, they are fully supervised with the initial 3D pseudo-GTs and weakly supervised with GT 2D/3D joint coordinates for the final 3D pseudo-GTs.

In the current literature, optimization-based annotators, such as SMPLify-X [24], are widely used. However, they often produce wrong 3D pseudo-GTs as they fit the 3D parameters to each sample independently. To overcome the limitation, we present NeuralAnnot, the first neural network-based one-stage annotator. The main idea of NeuralAnnot is to employ a neural network-based regressor and dedicate it for the annotation. Assuming no 3D pseudo-GTs are available, NeuralAnnot is weakly supervised with GT 2D/3D joint coordinates of training sets. Then, it is tested on the same training set. The testing results on the testing set become 3D pseudo-GTs, used to fully supervise the regressors. We train and test NeuralAnnot in the pre-processing stage of the regressors. NeuralAnnot's training and testing pipeline is clearly different from that of the regressors as the regressors are trained on training sets and tested on unseen test sets. In contrast, we train and test NeuralAnnot on the same training set. As it is tested on seen samples of the training set, there is no concern on the generalization performance on unseen data. Like SMPLify-X, our NeuralAnnot is a one-stage annotator, which does not require initial 3D pseudo-GTs and produces 3D pseudo-GTs only from GT 2D/3D joint coordinates.

In our experiments, we observed that simply employing a regressor for the annotation does not result in high-quality 3D pseudo-GTs. In particular, in-the-wild datasets only provide GT 2D joint coordinates without 3Ds; therefore, overcoming the depth ambiguity is one of the major bottlenecks for the annotation of in-the-wild datasets. We investigate the best combinations of inputs and outputs for the annotation of in-the-wild datasets and MoCap datasets.

For a fair comparison, we pick SMPLify-X [24] for our main comparison target as it is a one-stage annotator like our NeuralAnnot and the most widely used one. We additionally compare ours with SPIN and EFT in the experimental section after changing NeuralAnnot to a two-stage annotator. All evaluation metrics measure how the produced 3D pseudo-GTs of training sets are accurate and beneficial. We show that our NeuralAnnot produces 3D pseudo-GTs with much higher quality than SMPLify-X. We use NeuralAnnot to obtain 3D pseudo-GTs of the human body, hands, face, and whole body. All our 3D pseudo-GTs will be publicly available.

Our contributions can be summarized as follows.

- We present NeuralAnnot, the first neural network-based one-stage annotator. NeuralAnnot is trained and tested on the same training to produce 3D pseudo-GTs, used to fully supervise the regressors.
- Our NeuralAnnot produces far more accurate and beneficial 3D pseudo-GTs than previous optimization-based annotators owing to the data-driven learning of neural networks.
- The newly obtained 3D pseudo-GTs will be publicly released. We believe the new 3D pseudo-GTs will be highly beneficial to the community.

2. Related works

Optimization-based 3D pseudo-GT annotators. The optimization-based annotators fit 3D human model parameters for target 2D/3D joint coordinates, 3D point clouds, or 3D scans, which become 3D pseudo-GTs. They fit 3D human model parameters to each sample without considering other samples. SMPLify [1] fits SMPL [16] to a given 2D joint coordinates by minimizing a 2D loss and several prior terms. Joo *et al*. [8] constructed the Total Capture dataset by fitting a whole-body 3D human model (*i.e.*, Adam and Frank) to estimated 3D joint coordinates and 3D point clouds. SMPLify-X [24] extended SMPLify for the 3D human whole-body model optimization by proposing a new 3D human whole-body model, SMPL-X. Von *et al*. [28] constructed 3DPW dataset by fitting SMPL parameters to detected 2D joint coordinates and IMU sensor values. Zimmermann *et al*. [31] constructed FreiHAND dataset by fitting MANO parameters to multi-view 2D hand joint coordinates, 3D hand joint coordinates, and segmentations. Kulon *et al.* [13] constructed Youtube3DHand dataset by fitting MANO parameters to detected 2D hand joint coordinates. Patel *et al.* [23] constructed AGORA dataset by fitting SMPL and SMPL-X parameters to 3D scans. Among them, SMPLify-X [24] is the most widely used annotator to obtain 3D pseudo-GTs.

Neural network-based 3D pseudo-GT annotators. SPIN [11] and EFT [7] are recently introduced neural network-based annotators. SPIN first predicts SMPL parameters using a neural network and runs an optimization-based annotator [1], which fits the predicted SMPL parameters to GT 2D joint coordinates. Their final 3D pseudo-GT of each sample is obtained by selecting one with smaller SMPLify loss [1] between the outputs of their optimization-based annotator and prepared initial 3D pseudo-GTs. The initial 3D pseudo-GTs are prepared before training their network by running SMPLify on GT 2D joint coordinates. EFT [7] fine-tunes the pre-trained network of SPIN to the GT 2D joint coordinates of each sample, and the outputs of the last fine-tuning iteration become 3D pseudo-GT of the sample.

One critical difference between our NeuralAnnot and SPIN/EFT is that ours is a one-stage annotator, while SPIN/EFT are two-stage annotators. The one-stage annotators annotate 3D pseudo-GTs by being weakly supervised with GT 2D/3D joint coordinates without requiring initial 3D pseudo-GTs. On the other hand, the two-stage annotators obtain initial 3D pseudo-GTs using a one-stage annotator in the pre-processing stage of the two-stage annotators. SPIN and EFT use optimization-based annotators [1, 24] as an one-stage annotator to obtain the initial 3D pseudo-GTs. Then, they are fully supervised with the initial 3D pseudo-GTs and weakly supervised with GT 2D/3D joint coordinates in the second stage to obtain the final 3D pseudo-GTs. Due to the difference between the one-stage and two-stage annotators, SPIN and EFT are not our direct comparison targets. Instead, our NeuralAnnot can improve their annotation pipeline by replacing the optimization-based annotators in their pre-processing stage. Nevertheless, to show the clear benefit of NeuralAnnot, we compare ours with SPIN and EFT in the experimental section after changing NeuralAnnot to a two-stage annotator. The comparison shows that the two-stage NeuralAnnot produces much more beneficial 3D pseudo-GTs than SPIN and EFT.

3. NeuralAnnot

3.1. Network architecture

NeuralAnnot predicts 3D human model parameters Θ from a single image $\mathbf{I}$. We design NeuralAnnot as a simple combination of ResNet-50 [3] and a fully-connected layer

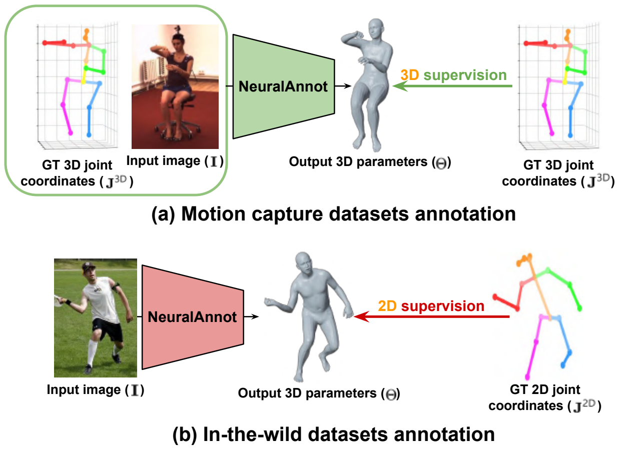

GT 3D joint coordinates ($\mathbf{J}^{3D}$)
Input image ($\mathbf{I}$)
NeuralAnnot
Output 3D parameters (Θ)
3D supervision
GT 3D joint coordinates ($\mathbf{J}^{3D}$)
(a) Motion capture datasets annotation
Input image ($\mathbf{I}$)
NeuralAnnot
Output 3D parameters (Θ)
2D supervision
GT 2D joint coordinates ($\mathbf{J}^{2D}$)
(b) In-the-wild datasets annotation

Figure 1. The overall pipeline of NeuralAnnot. (a) For the MoCap dataset annotation, it takes a pair of (image, GT 3D joint coordinates) and is supervised with GT 3D joint coordinates. (b) For the in-the-wild dataset annotation, it takes an image and is supervised with GT 2D joint coordinates.

f, which extracts image feature vector $\mathbf{f}$ and regresses a set of 3D human model parameters Θ, respectively. The 3D human model parameter set Θ is different for each human model, which will be described in Section 4. As our main focus is not proposing a new network architecture, we use the most widely used network architecture [9, 11], a combination of ResNet and fully-connected layers.

3.2. MoCap datasets annotation

The MoCap datasets provide images with monotonous appearances and GT 3D joint coordinates $\mathbf{J}^{3D}$. NeuralAnnot obtains 3D pseudo-GTs of MoCap datasets from scratch without initial 3D pseudo-GTs, which we call the one-stage annotation. To this end, we provide the image feauture vector $\mathbf{f}$ and GT 3D joint coordinates $\mathbf{J}^{3D}$ to our network, as shown in Figure 1 (a). Please note that 3D pseudo-GTs consist of 3D joint *rotations* and latent codes of 3D human models, which are not directly obtainable from 3D joint *coordinates*. The image feature vector $\mathbf{f}$ provides human articulation and shape information, while GT 3D joint coordinates $\mathbf{J}^{3D}$ additionally provide depth and real scale information, which 2D image features lack. To this end, the GT 3D joint coordinates $\mathbf{J}^{3D}$ are converted to a 512-dimensional feature by two fully connected layers and concatenated with global average pooled ResNet output feature $\mathbf{f}$. The concatenated feature is fed to a fully connected layer f for the 3D human model parameter regression. The 3D rotations of joints in the predicted 3D human model parameters are initially predicted in a 6D rotational representation of Zhou *et al.* [30] and converted to a 3D axis-angle rotation. The loss function is defined as follows:

$$L_{\text{mocap}} = \|\hat{\mathbf{J}}^{3D} - \mathbf{J}^{3D}\|_1 + \sum_{\theta \in \Theta} \lambda_{\text{mocap},\theta} \hat{\theta}^2, \tag{1}$$

where the hat mark indicates a predicted output. The first term is a weak supervision from GT 3D joint coordinates, and the second term is a regularizer. $\hat{\mathbf{J}}^{\text{3D}}$ is obtained by multiplying a joint regression matrix of a human model to a 3D mesh, where the 3D mesh is obtained by forwarding the 3D human model parameters Θ to a 3D human model layer. $\lambda_{\text{mocap},\theta}$ denotes $L2$ norm regularizer weight of each 3D human parameter θ. The $L2$ norm regularizer prevents implausible 3D human mesh, widely used in previous works [1,24]. The network's testing outputs Θ on the training set become the 3D pseudo-GTs.

3.3. In-the-wild datasets annotation

The in-the-wild datasets provide images with diverse appearances and GT 2D joint coordinates $\mathbf{J}^{\text{2D}}$. NeuralAnnot obtains 3D pseudo-GTs of in-the-wild datasets from scratch without initial 3D pseudo-GTs, which we call the one-stage annotation. To this end, a fully-connected layer f takes the image feature vector $\mathbf{f}$ and outputs 3D human model parameters Θ, as shown in Figure 1 (b). Different from the MoCap datasets annotation scenario, NeuralAnnot takes a single image as an input without additional joint coordinates, of which the reason is reported in the experimental section. As the in-the-wild datasets provide only GT 2D joint coordinates without 3D data, only 2D supervision is allowed when training NeuralAnnot's network, which can lead to implausible 3D human mesh due to the depth ambiguity. To prevent NeuralAnnot's network from generating implausible 3D human meshes, we design it to predict a low-dimensional embedding of the 3D rotations (*i.e.*, a latent code of VPoser [24] for the body part and PCA coefficients for the hand part [25]), following SMPLify-X [24], unlike directly predicting 3D joint rotations when trained on MoCap datasets. The low-dimensional embedding can effectively limit the output space to plausible 3D human articulation space, thus can prevent implausible 3D human mesh. The loss function is defined as follows:

$$L_{\text{wild}} = \|\hat{\mathbf{J}}^{\text{2D}} - \mathbf{J}^{\text{2D}}\|_1 + \sum_{\theta\in\Theta} \lambda_{\text{wild},\theta}\hat{\theta}^2, \tag{2}$$

where the hat mark indicates a predicted output. The first term is a weak supervision from GT 2D joint coordinates, and the second term is a regularizer. $\hat{\mathbf{J}}^{\text{2D}}$ is obtained by projecting 3D joint coordinates $\hat{\mathbf{J}}^{\text{3D}}$ to the image plane with predicted camera parameters. $\lambda_{\text{wild},\theta}$ denotes $L2$ norm regularizer weight of each 3D human model parameter θ. Like when training our network on MoCap datasets, we used $L2$ norm regularizer to prevent implausible 3D human mesh. The network's testing outputs Θ on the training set become the 3D pseudo-GTs.

4. 3D pseudo-GTs of body, hands, and face

We use NeuralAnnot to obtain the 3D body, hands, face, and whole-body pseudo-GTs. We provide how to obtain those 3D pseudo-GTs below.

Body. We use SMPL [16] and SMPL-X [24] as 3D body models. To obtain SMPL 3D pseudo-GTs, our network predicts 3D body global rotation $\theta_b^g \in \mathbb{R}^3$, 3D body rotations $\theta_b \in \mathbb{R}^{21\times3}$, shape parameter $\beta_b \in \mathbb{R}^{10}$, and camera parameter $k_b \in \mathbb{R}^3$. Our network for SMPL-X 3D pseudo-GTs predicts the same outputs; however, a joint set for 3D body rotation θ_b changes to that of SMPL-X. Other SMPL-X parameters from hands and face are set to zero. We set Θ to $\{\theta_b, \beta_b\}$ and $\{z_b, \beta_b\}$ when training NeuralAnnot's network for the MoCap datasets annotation and in-the-wild datasets annotation, respectively. z_b denotes the latent code of the VPoser [24].

Hands. We use MANO [25] as a 3D hand model. To obtain MANO 3D pseudo-GTs, our network predicts 3D hand global rotation $\theta_h^g \in \mathbb{R}^3$, 3D hand rotations $\theta_h \in \mathbb{R}^{15\times3}$, shape parameter $\beta_h \in \mathbb{R}^{10}$, and camera parameter $k_h \in \mathbb{R}^3$. All hand images are flipped to the right hands, and we flip back the estimated pseudo-GTs of the left hands. We set Θ to $\{\theta_h, \beta_h\}$ and $\{z_h, \beta_h\}$ when training NeuralAnnot's network for the MoCap datasets annotation and in-the-wild datasets annotation, respectively. z_h denotes the 3D hand pose PCA coefficients, defined in MANO.

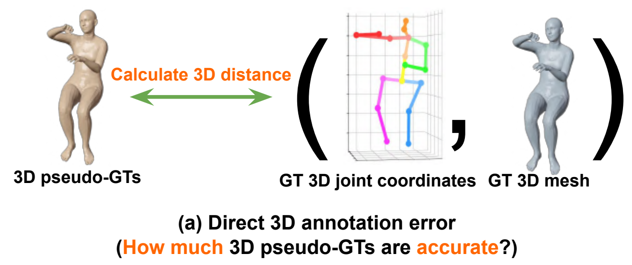
Face. We use FLAME [14] as a 3D face model. To obtain FLAME 3D pseudo-GTs, our network predicts 3D face global rotation $\theta_f^g \in \mathbb{R}^3$, 3D jaw rotation $\theta_f \in \mathbb{R}^3$, shape parameter $\beta_f \in \mathbb{R}^{10}$, and expression code $\psi \in \mathbb{R}^{10}$. We set Θ to $\{\theta_f, \beta_f, \psi\}$.

Integration to whole body. After obtaining the body, hands, and face 3D pseudo-GTs, we get the final whole-body 3D pseudo-GTs by forwarding $\{\theta_b^g, \theta_b, \beta_b, \theta_{rh}^g, \theta_{rh}, \theta_{lh}^g, \theta_{lh}, \theta_f, \psi\}$ to SMPL-X, where $*_{rh}$ and $*_{lh}$ denote $*$ is from right and left hand, respectively. θ_b^g, θ_b, and β_b are from SMPL-X 3D pseudo-GTs. The 3D hand rotations θ_h of MANO and 3D jaw rotation θ_f and facial expression code ψ of FLAME are compatible with those of SMPL-X; thus, we use them for the final 3D pseudo-GTs. As θ_h^g are not 3D local rotations in the human body kinematic chain, we change their 3D global rotations to 3D local rotations by multiplying the inverse of 3D elbow global rotations.

5. Experiment

5.1. Implementation details

PyTorch [22] is used for implementation. The backbone part is initialized with the publicly released ResNet50 [3], pre-trained on ImageNet [27]. The weights are updated by the Adam optimizer [10] with a mini-batch size of 192. All input images are cropped using GT box and resized to 256×256. The initial learning rate is set to 10^{-4} and reduced by a factor of 10 when it converges.


Calculate 3D distance
3D pseudo-GTs
GT 3D joint coordinates
GT 3D mesh
(a) Direct 3D annotation error
(How much 3D pseudo-GTs are accurate?)

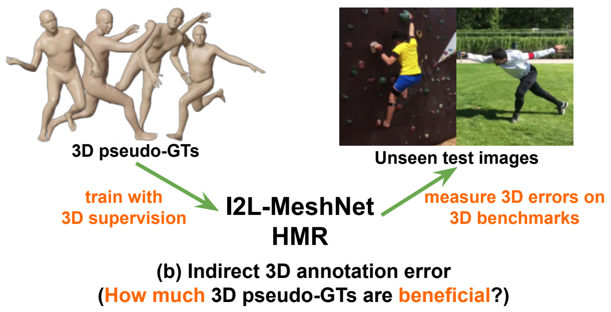


Figure 2. Illustrations of our metrics to evaluate annotators. (a) Direct 3D annotation error measures MPJPE and MPVPE, only applicable to datasets that provide 3D GTs. (b) Indirect 3D annotation error measures 3D errors of regressors, such as I2L-MeshNet [20] and HMR [9], on 3D benchmarks after training them on the obtained 3D pseudo-GTs of MSCOCO. This is used for in-the-wild datasets as the absence of 3D GTs makes it impossible to calculate the direct 3D annotation error.

5.2. Datasets for the annotation

We use various datasets for the annotation as described below. Please note that in addition to the datasets below, our NeuralAnnot can be used for any datasets that provide GT 2D/3D joint coordinates.

MoCap datasets. We annotate 3D body pseudo-GTs of Human3.6M [4], MPI-INF-3DHP [17], and 3DPW [28]. Also, 3D hand pseudo-GTs of InterHand2.6M [21] and FreiHAND [31] are annotated. Among them, only 3DPW and FreiHAND have GT 3D meshes, and all of them have only GT 3D joint coordinates without meshes.

In-the-wild datasets. We annotate 3D body, hands, face, and whole-body pseudo-GTs of MSCOCO [5, 15]. It only provides GT 2D joint coordinates without 3D ones.

5.3. Evaluation metrics

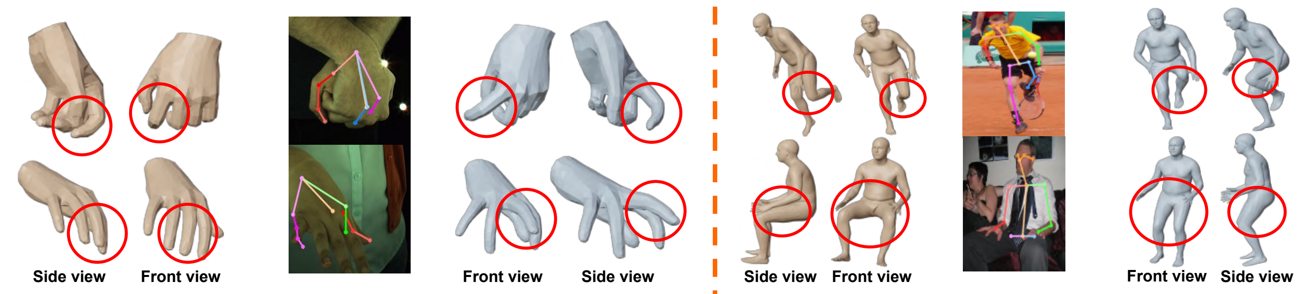
Direct 3D annotation error. Figure 2 (a) shows how we calculate the direct 3D annotation error. The direct 3D annotation error is measured using mean per joint position error (MPJPE) and mean per-vertex position error (MPVPE), which are the average 3D joint distance (mm) and 3D mesh vertex distance (mm) between GT and ones from 3D pseudo-GTs after aligning a root joint position. This metric is only applicable to datasets that provide GT 3D joint coordinates or GT 3D meshes.

Indirect 3D annotation error. Figure 2 (b) shows how we designed the indirect 3D annotation error. The indirect 3D annotation error represents how much the 3D pseudo-GTs are beneficial for the training. The absence of the 3D GTs of the in-the-wild datasets makes calculating the direct 3D annotation error impossible. Alternatively, we use 3D test errors of regressors after training them on 3D pseudo-GTs of MSCOCO as an indirect 3D annotation error. The indirect 3D annotation error would decrease if the 3D pseudo-GTS from annotators become more beneficial so that the trained regressors work better. We use I2L-MeshMet [20] and HMR [9] as regressors for the indirect error calculation. I2L-MeshNet and HMR are based on model-free and model-based approaches, which predict 3D vertex coordinates and 3D human model parameters, respectively. We chose such very different regressors to minimize dependency on the type of the regressors so that the indirect error could be less stochastic. For the indirect error of the body part, we use PA MPJPE (mm) and PA MPVPE (mm), which further align rotation and scale in addition to the translation. For the hand part, area under curve (AUC) of a 3D mesh is used. Finally, for the face part, a 3D surface error, an average distance between GT 3D scan and the closest 3D mesh vertex is used.

5.4. Ablation study

Inputs of NeuralAnnot. For the MoCap datasets annotation, our network takes a pair of (image, GT 3D joint coordinates) as an input. The Figure 3 left (a-c) and first, second, and third rows of Table 2 validate our input choice for the MoCap datasets annotation. The figure shows that taking an image as input is greatly helpful to recover partially missing GT 3D joint coordinates, which arises from the failure of the marker-less data capture. The table shows that additional GT 3D joint coordinates reduce the direct 3D error by providing depth and scale information to the network.

For the in-the-wild datasets annotation, our network takes an image as an input. The Figure 3 right (d-f) and the fourth, fifth, and sixth rows of Table 2 validate our input choice for the in-the-wild datasets annotation. The figure shows that taking an image suffers from less depth ambiguity as image appearances can provide depth information, which lacks in 2D joint coordinates. For example, in the first row of the figure, the shadow and small size of the left knee tell us that the leg is behind the right leg. In addition, the image input can provide contextual information, helpful when GT 2D joint coordinates are missing or truncated. The table shows that 3D pseudo-GTs from the image input achieve lower indirect 3D annotation error. We observed that using both an image and GT 2D joint coordinates as an input achieves similar indirect 3D error compared with a



(a) From GT 3D joint coordinates + image (Ours) (b) Input image overlaid with GT joint coordinates (c) From GT 3D joint coordinates (d) From an image (Ours) (e) Input image overlaid with GT joint coordinates (f) From GT 2D joint coordinates

Figure 3. Qualitative comparison of 3D hand and body pseudo-GTs from NeuralAnnots that take various inputs on InterHand2.6M (left, a-c) and MSCOCO (right, d-f). The top row of results on InterHand2.6M is from a left hand, occluded by the right hand. GT 3D joint coordinates of the samples from InterHand2.6M are partially missing due to the capture failure. GT 2D joint coordinates of the bottom sample from MSCOCO are partially missing due to the truncation.

Inputs of NeuralAnnot	Annotation err.
*** MoCap datasets annotation**	**Direct 3D err.**
Image	7.0
GT 3D joint coordinates	8.1
GT 3D joint coordinates + Image (Ours)	**5.8**
*** In-the-wild datasets annotation**	**Indirect 3D err.**
Image (Ours)	**69.6**
GT 2D joint coordinates	72.7
GT 2D joint coordinates + Image	**69.6**

Table 2. The annotation error comparison of NeuralAnnots that take different inputs. The direct 3D errors (MPJPE) are computed on InterHand2.6M, and the indirect 3D errors (PA MPJPE) are computed on 3DPW after training I2L-MeshNet on 3D pseudo-GTs of MSCOCO.

Outputs of NeuralAnnot	Annotation err.
*** MoCap datasets annotation**	**Direct 3D err.**
3D joint rotations (Ours)	**5.8**
Low-dimensional embedded pose	10.7
*** In-the-wild datasets annotation**	**Indirect 3D err.**
3D joint rotations	117.5
Low-dimensional embedded pose (Ours)	**69.6**

Table 3. The annotation error comparison of NeuralAnnots that produce different outputs. The direct 3D errors (MPJPE) are computed on InterHand2.6M, and the indirect 3D errors (PA MPJPE) are computed on 3DPW after training I2L-MeshNet on 3D pseudo-GTs of MSCOCO.

model that takes an image as an input, while raising computational costs. We think the reason for the similar indirect 3D error is that the GT 2D joint coordinates do not offer additional depth information as GT 3D joint coordinates did in the MoCap datasets. To achieve the best quality of 3D pseudo-GTs with smaller computational costs, we used only an image as an input of the network without GT 2D joint coordinates. For the comparison, we provided GT 2D joint coordinates as a 2D Gaussian heatmap representation to our network following Moon *et al.* [18].

Outputs of NeuralAnnot. Table 3 validates our design for NeuralAnnot's output type. For the MoCap datasets annotation, our network outputs 3D joint rotations while it outputs the low-dimensional embedded poses (*e.g.*, VPoser of SMPL/SMPL-X and PCA coefficients of MANO) for the in-the-wild datasets annotation. The low-dimensional embedded poses can be helpful when GT carries insufficient 3D information (*e.g.*, 2D joint coordinates) by restricting outputs to learned latent pose space. For example, it can prevent many anatomically implausible 3D meshes that correspond to 2D joint coordinates. On the other hand, the low-dimensional embedded poses can be harmful when GT carries enough 3D information (*e.g.*, 3D joint coordinates) as the 3D information of GT is enough to prevent the implausible 3D meshes, while some 3D poses cannot be represented in the learned latent pose space. The first and second rows of the table show that when annotating MoCap datasets, estimating 3D joint rotations achieves better results due to the availability of GT 3D joint coordinates. The third and fourth rows of the table show that when annotating in-the-wild datasets, estimating the low-dimensional embedded pose produces better results as only GT 2D joint coordinates are available without 3D ones. We designed the network to produce the best output type considering which type of supervision is available for the annotation of each dataset.

5.5. Comparison with previous annotators

As described in Section 1, for the fair comparison, we pick SMPLify-X [24] as our main comparison target as it is a one-stage annotator like our NeuralAnnot and the most widely used one. We additionally compare ours with two-stage annotators, such as SPIN [11] and EFT [7], after changing NeuralAnnot to the two-stage annotator.

MoCap datasets annotation using GT 3D joint coordinates. Table 4 shows that NeuralAnnot achieves much

Datasets	SMPLify-X	**NeuralAnnot (Ours)**
* **Body**		
Human3.6M	13.1 / N/A	**8.5** / N/A
MPI-INF-3DHP	17.9 / N/A	**12.2** / N/A
3DPW	19.2 / 38.0	**10.7 / 11.1**
* **Hands**		
InterHand2.6M	10.0 / N/A	**5.8** / N/A
FreiHAND	6.6 / 7.3	**3.6 / 4.0**

Table 4. The direct 3D annotation error (MPJPE/MPVPE) comparison between SMPLify-X and our NeuralAnnot on various MoCap datasets. **The 3D pseudo-GTs are obtained by being fit to GT 3D joint coordinates.**

lower direct 3D annotation error than SMPLify-X [24] on various MoCap datasets when the 3D pseudo-GTs are fit to GT 3D joint coordinates like Figure 1 (a). GT 3D meshes are not used for the annotation but used only for evaluation purposes. Moreover, NeuralAnnot can recover 3D pseudo-GTs from partially missing and noisy GT 3D joint coordinates by utilizing image features, as shown in Figure 3 left (a-c), while SMPLify-X cannot as it only utilizes GT 3D joint coordinates without the image input. For the comparison, we modified official released codes of SMPLify-X to optimize it to GT 3D joint coordinates as the original one only considers 2D joint coordinates during the optimization. Like NeuralAnnot, SMPLify-X optimizes 3D joint rotations, not low-dimensional embedded poses, for the MoCap datasets annotation. We describe its detailed modifications in the supplementary material. MPVPEs are reported only for datasets that provide GT 3D meshes.

MoCap datasets annotation using GT 2D joint coordinates. Table 5 shows that NeuralAnnot achieves much lower direct 3D annotation error than SMPLify-X [24] on various MoCap datasets when the 3D pseudo-GTs are fit to GT 2D joint coordinates. The table is to simulate an in-the-wild dataset annotation scenario, in which only GT 2D joint coordinates are available, and provide direct 3D annotation errors. We annotate 3DPW and FreiHAND datasets as they contain diverse outdoor images like in-the-wild images of MSCOCO compared to other MoCap datasets, such as Human3.6M and InterHand2.6M To this end, we use the pipeline of Figure 1 (b) for the MoCap dataset annotation. SMPLify-X fits 3D human model parameters to GT 2D joint coordinates. NeuralAnnot takes a single image from a MoCap dataset and supervises the network with GT 2D joint coordinates. MPVPEs are reported only for datasets that provide GT 3D meshes.

In-the-wild datasets annotation using GT 2D joint coordinates. Table 6 shows that 3D pseudo-GTs from our NeuralAnnot achieve lower indirect 3D annotation error than SMPLify-X when the 3D pseudo-GTs are fit to GT 2D joint coordinates like Figure 1 (b). For the comparison, we ran

Datasets	SMPLify-X	**NeuralAnnot (Ours)**
* **Body**		
3DPW	147.35 / 206.62	**60.1 / 68.3**
* **Hands**		
FreiHAND	8.5 / 9.3	**7.2 / 7.7**

Table 5. The direct 3D annotation error (MPJPE/MPVPE) comparison between SMPLify-X and our NeuralAnnot on various MoCap datasets. **The 3D pseudo-GTs are obtained by being fit to GT 2D joint coordinates.**

Annotators	Indirect 3D err.
* **Body**	**PA MPVPE↓**
SMPLify-X	80.7 [20] / 81.4 [9]
NeuralAnnot (Ours)	**71.6** [20] / **73.3** [9]
* **Hands**	**AUC of 3D mesh↑**
SMPLify-X	0.731 [20] / 0.755 [9]
NeuralAnnot (Ours)	**0.760** [20] / **0.786** [9]
* **Face**	**3D surface err.↓**
SMPLify-X	2.05 [20] / 2.06 [9]
NeuralAnnot (Ours)	**2.03** [20] / **2.05** [9]

Table 6. The indirect 3D annotation errors comparison with one-stage annotator, SMPLify-X. The indirect 3D annotations errors are computed on 3DPW, FreiHAND, and Stirling for the body, hands and face, respectively, after training I2L-MeshNet [20]/HMR [9] on 3D pseudo-GTs of MSCOCO.

Annotators	Indirect 3D err.
SPIN	77.4 [20] / 80.1 [9]
EFT	71.2 [20] / 72.1 [9]
Two-stage NeuralAnnot (Ours)	**64.6** [20] / **65.1** [9]

Table 7. The indirect 3D annotation error (PA MPVPE) comparison with two-stage annotators. The indirect 3D annotation errors are computed on 3DPW after training I2L-MeshNet [20]/HMR [9] on 3D pseudo-GTs of MSCOCO.

SMPLify-X using their officially released codes. It optimizes the low-dimensional embedded poses for the in-the-wild datasets annotation, like ours. Figure 4 shows that SMPLify-X fails when the target image's pose has high depth ambiguity or there are truncations. Unlike the body and hands, NeuralAnnot produces similar indirect 3D annotation error of 3D face pseudo-GTs compared with those of SMPLify-X. We think the reason is that less complicated articulations of the face make SMPLify-X work well.

We further show the superiority of NeuralAnnot in Table 7 by comparing it with the two-stage annotators, such as SPIN and EFT. To this end, we change NeuralAnnot to a two-stage annotator by jointly training it on 3D pseudo-GTs of MoCap datasets (*i.e.*, Human3.6M and MPI-INF-3DHP) and in-the-wild datasets (*i.e.*, MSCOCO), where the 3D pseudo-GTs are obtained from our original one-stage NeuralAnnot. For the results of SPIN and EFT, we used

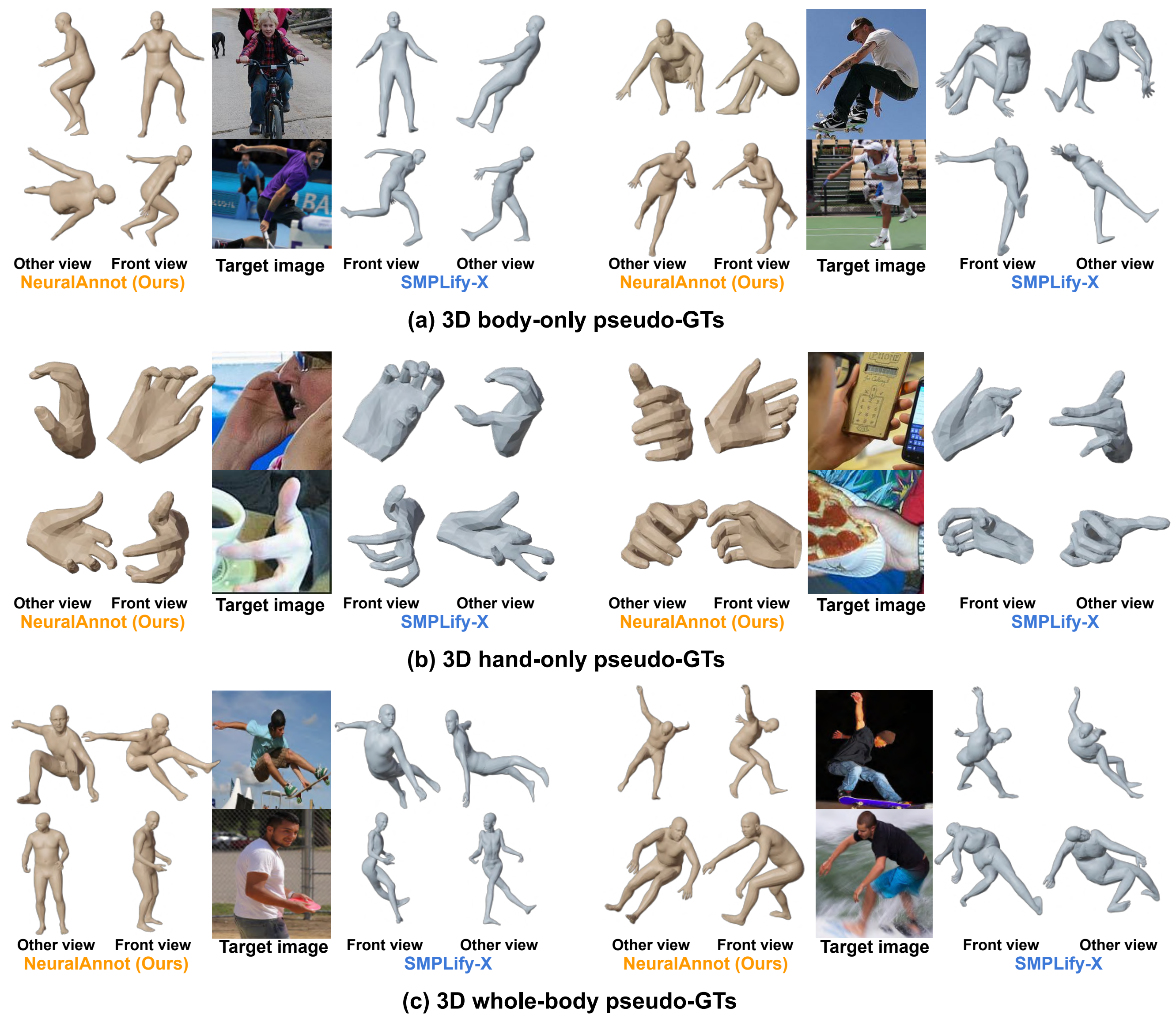

Other view
Front view
NeuralAnnot (Ours)
Target image
Front view
Other view
SMPLify-X
(a) 3D body-only pseudo-GTs
(b) 3D hand-only pseudo-GTs
(c) 3D whole-body pseudo-GTs

Figure 4. Qualitative comparison of 3D pseudo-GTs from NeuralAnnot and SMPLify-X.

their officially released 3D pseudo-GTs. The comparisons show that NeuralAnnot produces 3D pseudo-GTs with the lowest indirect 3D annotation error. We think this is because our initial 3D pseudo-GTs, obtained by our original one-stage NeuralAnnot, are much beneficial than those of SPIN and EFT, obtained by the optimization-based annotators.

6. Conclusion

We present NeuralAnnot, the first neural network-based one-stage annotator. NeuralAnnot is the first attempt to dedicate a neural network for the 3D human model parameter annotation. It is trained and tested on the same training set with GT 2D/3D joint coordinates in the pre-processing stage of 3D human mesh regressors. After the pre-processing stage, the outputs of NeuralAnnot on the training set become 3D pseudo-GTs, used to train the regressors. NeuralAnnot produces much more accurate and beneficial 3D pseudo-GTs than the most widely used one-stage annotator, SMPLify-X. We will release our 3D pseudo-GTs, which will be highly beneficial to the community.

Acknowledgements. This work was supported in part by IITP grant funded by the Korea government (MSIT) [No. 2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)]

References

[1] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *ECCV*, 2016. 2, 3, 4

[2] Vasileios Choutas, Georgios Pavlakos, Timo Bolkart, Dimitrios Tzionas, and Michael J Black. Monocular expressive body regression through body-driven attention. In *ECCV*, 2020. 1, 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 4

[4] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *TPAMI*, 2014. 1, 5

[5] Sheng Jin, Lumin Xu, Jin Xu, Can Wang, Wentao Liu, Chen Qian, Wanli Ouyang, and Ping Luo. Whole-body human pose estimation in the wild. In *ECCV*, 2020. 1, 5

[6] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic Studio: A massively multiview system for social motion capture. In *ICCV*, 2015. 1

[7] Hanbyul Joo, Natalia Neverova, and Andrea Vedaldi. Exemplar fine-tuning for 3D human pose fitting towards in-the-wild 3D human pose estimation. *arXiv preprint arXiv:2004.03686*, 2020. 1, 2, 3, 6

[8] Hanbyul Joo, Tomas Simon, and Yaser Sheikh. Total Capture: A 3D deformation model for tracking faces, hands, and bodies. In *CVPR*, 2018. 2

[9] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018. 1, 2, 3, 5, 7

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 4

[11] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3D human pose and shape via model-fitting in the loop. In *ICCV*, 2019. 1, 2, 3, 6

[12] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *CVPR*, 2019. 1

[13] Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael M Bronstein, and Stefanos Zafeiriou. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *CVPR*, 2020. 3

[14] Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM TOG*, 2017. 1, 4

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1, 5

[16] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. SMPL: A skinned multi-person linear model. *ACM TOG*, 2015. 1, 2, 4

[17] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3D human pose estimation in the wild using improved cnn supervision. In *3DV*, 2017. 1, 5

[18] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. PoseFix: Model-agnostic general human pose refinement network. In *CVPR*, 2019. 6

[19] Gyeongsik Moon, Hongsuk Choi, and Kyoung Mu Lee. Accurate 3D hand pose estimation for whole-body 3D human mesh estimation. In *CVPRW*, 2022. 1

[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. In *ECCV*, 2020. 1, 2, 5, 7

[21] 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, 5

[22] 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. 4

[23] Priyanka Patel, Chun-Hao P. Huang, Joachim Tesch, David T. Hoffmann, Shashank Tripathi, and Michael J. Black. AGORA: Avatars in geography optimized for regression analysis. In *CVPR*, 2021. 3

[24] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, 2019. 1, 2, 3, 4, 6, 7

[25] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied Hands: Modeling and capturing hands and bodies together. *ACM TOG*, 2017. 1, 4

[26] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. FrankMocap: A monocular 3d whole-body pose estimation system via regression and integration. In *ICCV Workshop*, 2021. 1, 2

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 4

[28] Timo von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3D human pose in the wild using IMUs and a moving camera. In *ECCV*, 2018. 2, 5

[29] Zhixuan Yu, Jae Shin Yoon, In Kyu Lee, Prashanth Venkatesh, Jaesik Park, Jihun Yu, and Hyun Soo Park. HUMBI: A large multiview dataset of human body expressions. In *CVPR*, 2020. 1

[30] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019. 3

[31] 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 *ICCV*, 2019. 2, 5