DeepHandMesh: A Weakly-Supervised Deep Encoder-Decoder Framework for High-Fidelity Hand Mesh Modeling

Gyeongsik Moon[1], Takaaki Shiratori[2], and Kyoung Mu Lee[1]

[1] ECE & ASRI, Seoul National University, Korea
[2] Facebook Reality Labs
{mks0601,kyoungmu}@snu.ac.kr, tshiratori@fb.com

Abstract. Human hands play a central role in interacting with other people and objects. For realistic replication of such hand motions, high-fidelity hand meshes have to be reconstructed. In this study, we firstly propose DeepHandMesh, a weakly-supervised deep encoder-decoder framework for high-fidelity hand mesh modeling. We design our system to be trained in an end-to-end and weakly-supervised manner; therefore, it does not require groundtruth meshes. Instead, it relies on weaker supervisions such as 3D joint coordinates and multi-view depth maps, which are easier to get than groundtruth meshes and do not dependent on the mesh topology. Although the proposed DeepHandMesh is trained in a weakly-supervised way, it provides significantly more realistic hand mesh than previous fully-supervised hand models. Our newly introduced penetration avoidance loss further improves results by replicating physical interaction between hand parts. Finally, we demonstrate that our system can also be applied successfully to the 3D hand mesh estimation from general images. Our hand model, dataset, and codes are publicly available[1].

1 Introduction

Social interactions are vital to humans: every day, we spend a large amount of time on interactions and communications with other people. While facial motion and speech play a central role in communication, important non-verbal information is also communicated via body motion, especially hand and finger motion, to emphasize our speech, clarify our ideas, and convey emotions. Modeling and replicating detailed hand geometry and motion is essential to enrich experience in various applications, including remote communications in virtual/augmented reality and digital storytelling such as movies and video games.

A pioneering work of hand geometry modeling is MANO by Romero et al. [28], which consists of linear models of identity- and pose-dependent correctives with linear blend skinning (LBS) as an underlying mesh deformation algorithm. The model is learned in a fully-supervised manner by minimizing the

[1] https://mks0601.github.io/DeepHandMesh/

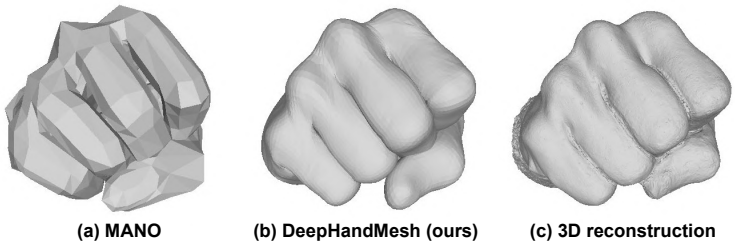

Fig. 1: Qualitative result comparison between (a) MANO [28], (b) our DeepHandMesh, and (c) 3D reconstruction [5].

per-vertex distance between output and groundtruth meshes that are obtained by registering a template mesh to 3D hand scans [2, 11].

Although MANO has been widely used for hand pose and geometry estimation [1, 3, 9], there exist limitations. First, their method requires groundtruth hand meshes (*i.e.*, the method requires per-vertex supervision to train the linear model). As the hand contains many self-occlusions and self-similarities, existing mesh registration methods [2, 11] sometimes fail. To obtain the best quality of groundtruth hand meshes, Romero et al. [28] manually inspected each registered mesh and discarded failed ones from the training data, which requires extensive manual labor. Second, its fidelity is limited. As MANO uses the hand parts of SMPL [21], its resolution is low (*i.e.*, 778 vertices). This low resolution could limit the expressiveness of the reconstructed hand meshes. Also, MANO consists of linear models, optimized by the classical optimization framework. As recent deep neural networks (DNNs) that consist of many non-linear modules show noticeable performance in many computer vision and graphics tasks, utilizing the DNNs with recent deep learning optimization techniques can give more robust and stable results. Finally, it does not consider physical interaction between hand parts. A model without consideration of the physical interaction could result in implausible hand deformation, such as penetration between hand parts.

In this paper, we firstly present *DeepHandMesh*, a weakly-supervised deep encoder-decoder framework for high-fidelity hand mesh modeling, that produces high-fidelity hand meshes from single images. Unlike existing methods such as MANO that require mesh registration for per-vertex supervision (*i.e.*, full supervision), DeepHandMesh utilizes only 3D joint coordinates and multi-view depth maps for supervision (*i.e.*, weak supervision). Therefore, our method avoids expensive data pre-processing such as registration and manual inspection. In addition, obtaining the 3D joint coordinates and depth maps is much easier compared with the mesh registration. The 3D joint coordinates can be obtained from powerful state-of-the-art multi-view 3D human pose estimation methods [18], and the depth maps can be rendered from 3D reconstruction [5] based on the solid mathematical theory about epipolar geometry. Furthermore, these are independent of topology of a hand model, allowing us to use hand meshes with

various topology and to be free from preparing topology-specific data such as registered meshes for each topology. To achieve high-fidelity hand meshes, DeepHandMesh is based on a DNN and optimized with recent deep learning optimization techniques, which provides more robust and stable results. We also use a high-resolution hand model to benefit from the expressiveness of the DNN. Our DeepHandMesh can replicate realistic hand meshes with details such as creases and skin bulging, as well as holistic hand poses. In addition, our newly designed penetration avoidance loss further improves results by enabling our system to replicate physical interaction between hand parts. Figure 1 shows that the proposed DeepHandMesh provides significantly more realistic hand meshes than the existing fully-supervised hand model (*i.e.*, MANO [28]).

As learning a high-fidelity hand model only via weak supervisions is a challenging problem, we assume a personalized environment (*i.e.*, assume the same subject in the training and testing stage). We discuss the limitations of the assumption and future research directions in the later section. To demonstrate the effectiveness of DeepHandMesh for practical purposes, we combine our DeepHandMesh with 3D pose estimation to build a model-based 3D hand mesh estimation system from a single image, as shown in Figure 2, and train it on a public dataset captured from general environments. The experimental results show that our DeepHandMesh can be applied to 3D high-fidelity hand mesh estimation from general images in real-time (*i.e.*, 50 fps).

Our contributions can be summarized as follows.

- We firstly propose a deep learning-based weakly-supervised encoder-decoder framework (DeepHandMesh) that is trained in an end-to-end, weakly-supervised manner for high-fidelity hand mesh modeling. Our proposed DeepHandMesh does not require labor-intensive manual intervention, such as mesh registration.
- Our weakly-supervised DeepHandMesh provides significantly more realistic hand meshes than previous fully-supervised hand models. In addition, we newly introduce a penetration avoidance loss, which can make DeepHandMesh firstly reproduce physical interaction between hand parts.
- We show that our framework can be applied to practical purposes, such as 3D hand mesh estimation from general images in real-time.

2 Related works

3D hand pose estimation. 3D hand pose estimation methods can be categorized into depth map-based and RGB-based ones according to their input. Early depth map-based methods are mainly based on a generative approach, which fits a pre-defined hand model to the input depth map by minimizing hand-crafted cost functions [30, 34] using particle swarm optimization [30], iterative closest point [33], or their combination [27]. Most of recent depth map-based methods are based on a discriminative approach, which directly localizes hand joints from an input depth map. Tompson et al. [35] utilized a neural network to localize

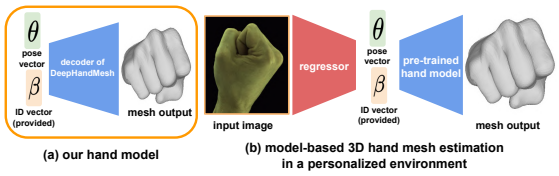

Fig. 2: (a) The hand model outputs meshes from the hand model parameters. Our main goal is to train a high-fidelity hand model in a weakly-supervised way. (b) The model-based 3D hand mesh estimation system outputs inputs of the hand model and use a pre-trained hand model to produce final hand meshes.

hand joints by estimating 2D heatmaps for each hand joint. Ge et al. [6] extended this method by estimating multi-view 2D heatmaps. Moon et al. [22] designed a 3D CNN that takes a voxel representation of a hand as input and outputs a 3D heatmap for each joint. Wan et al. [36] proposed a self-supervised system, which can be trained from only an input depth map.

The powerful performance of the recent CNN makes 3D hand pose estimation methods work well on RGB images. Zimmermann et al. [43] proposed a DNN that learns an implicit 3D articulation prior. Mueller et al. [24] used an image-to-image translation model to generate synthetic hand images for more effective training of a pose prediction model. Cai et al. [4] and Iqbal et al. [12] implicitly reconstruct depth map from an input RGB image and estimate 3D hand joint coordinates from it. Spurr et al. [32] and Yang et al. [38] proposed variational auto-encoders (VAEs) that learn a latent space of a hand skeleton and appearance.

3D hand shape estimation. Panteleris et al. [25] fitted a pre-defined hand model by minimizing reprojection errors of 2D joint locations w.r.t. hand landmarks detected by OpenPose [31]. Ge et al. [7] proposed a graph convolution-based network which directly estimates vertices of a hand mesh. Many recent methods are based on the MANO hand model. They train their new encoders and use a pre-trained MANO model as a decoder to generate hand meshes. Baek et al. [1] trained their network to estimate input vectors of the MANO model using neural renderer [14]. Boukhayma et al. [3] proposed a network that takes a single RGB image and estimates pose and shape vectors of MANO. Their network is trained by minimizing the distance of the estimated hand joint locations and groundtruth. Recently, Zimmermann et al. [44] proposed a marker-less captured 3D hand pose and mesh dataset.

3D hand model. MANO [28] is the most widely used hand model. It takes pose and shape vectors (*i.e.*, relative rotation of hand joint w.r.t. its parent joint and principal component analysis coefficients of hand shape space, respectively) as inputs and outputs deformed mesh using LBS and per-vertex correctives. It is trained from registered hand meshes in a fully-supervised way by minimizing the per-vertex distance between the output and the groundtruth hand meshes. Recently, Kulon et al. [17] proposed a hand model that takes a mesh latent code

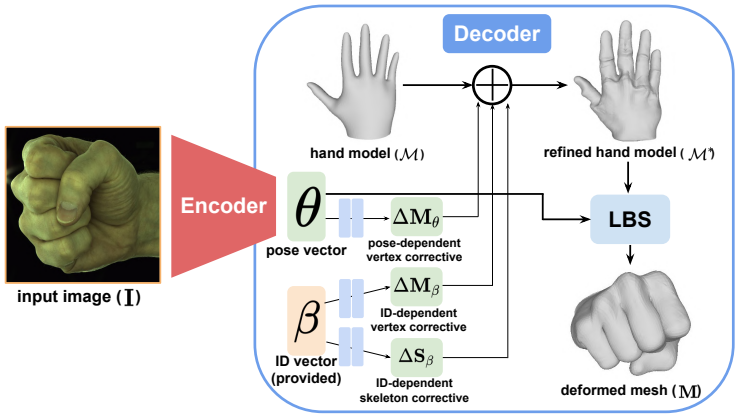

Fig. 3: Overall pipeline of the proposed DeepHandMesh.

and outputs a hand mesh using mesh convolution. To obtain the groundtruth meshes, they registered their new high-resolution hand model to 3D joint coordinates of Panoptic dome dataset [31]. They also compute a distribution of valid poses from the hand meshes registered to ∼1000 scans from the MANO dataset. They use this distribution to sample groundtruth hand meshes and train their hand model in a fully-supervised way using per-vertex mesh supervision.

All the above 3D hand models rely on mesh supervision (*i.e.*, trained by minimizing the per-vertex distance between output and groundtruth hand mesh) during training. In contrast, our DeepHandMesh is trained in a weakly-supervised setting, which does not require any groundtruth hand meshes. Although ours is trained without mesh supervision, it successfully reconstructs significantly more high-fidelity hand meshes, including creases and skin bulging, compared with previous hand models. Also, our DeepHandMesh is the first hand model that can replicate physical interaction between hand parts. This is a significant advancement compared with previous hand models.

3 Hand model

Our hand model is defined as $\mathcal{M} = \{\bar{\mathbf{M}}, \mathbf{S}; \mathbf{W}, \mathcal{H}\}$. $\bar{\mathbf{M}} = [\bar{\mathbf{m}}_1, \dots, \bar{\mathbf{m}}_V]^T \in \mathbb{R}^{V\times 3}$ denotes vertex coordinates of a zero-pose template hand mesh, where $\bar{\mathbf{m}}_v$ is 3D coordinates of vth vertex of $\bar{\mathbf{M}}$. V denotes the number of vertices. $\mathbf{S} \in \mathbb{R}^{J\times 3}$ means the translation vector of each hand joint from its parent joint, where J is the number of joints. $\mathbf{W} \in \mathbb{R}^{V\times J}$ denotes skinning weights for LBS. Finally, $\mathcal{H}$ denotes a hand joint hierarchy. Our template hand model is prepared by artists. The parameters on the right of the semicolon do not change during training. Thus, we omit them hereafter for simplicity.

4 Encoder

4.1 Hand pose vector

The encoder takes a single RGB image of a hand $\mathbf{I}$ and estimates its hand pose vector $\theta \in \mathbb{R}^{N_\mathrm{P}}$, where $N_\mathrm{P} = 28$ denotes the degrees of freedom (DOFs) of it. Among all the DOFs of the hand joint rotation $3J$, we selected N_P DOFs based on the prior knowledge of human hand anatomical property and the hand models of [40, 42]. For the enabled DOFs, the estimated hand pose vector is used as a relative Euler angle w.r.t. its parent joint. We set all the disabled DOFs to zero and fixed them during the optimization.

4.2 Network architecture

Our encoder consists of ResNet-50 [10] and two fully-connected layers. The ResNet extracts a hand image feature from the input RGB image $\mathbf{I}$. Then, the extracted feature is passed to the two fully-connected layers, which outputs the hand pose vector θ. The hidden activation size of the fully-connected layers is 512, and the ReLU activation function is used after the first fully-connected layer. To ensure θ in the range of (-π, π), we apply a hyperbolic tangent activation function at the output of the second fully-connected layer and multiply it by π.

5 Decoder

5.1 Hand model refinement

To replicate details on the hand model, we designed the decoder to estimate three correctives from a pre-defined identity vector $\beta \in \mathbb{R}^{N_\mathrm{I}}$ and an estimated hand pose vector θ, inspired by [21, 28], as shown in Figure 3. As the proposed DeepHandMesh assumes a personalized environment (*i.e.*, assumes the same subject in the training and testing stage), we pre-define β as a $N_\mathrm{I} = 32$ dimensional randomly initialized normal Gaussian vector for each subject. β is fixed during training and testing. Note that DeepHandMesh does not require a personalized hand model to be given. Rather, it personalizes an initial hand mesh for a training subject during training.

The first corrective is identity-dependent skeleton corrective $\Delta\mathbf{S}_\beta \in \mathbb{R}^{J\times 3}$. As hand shape and size vary for each person, 3D joint locations can be different for each person. To personalize $\mathbf{S}$ to a training subject, we build two fully-connected layers in our decoder and estimate $\Delta\mathbf{S}_\beta$ from the pre-defined identity code β. The hidden activation size of the fully-connected layer is 256. The estimated $\Delta\mathbf{S}_\beta$ is added to $\mathbf{S}$, yielding $\mathbf{S}^*$. Figure 4 (b) shows the effect of $\Delta\mathbf{S}_\beta$.

The second corrective is identity-dependent per-vertex corrective $\Delta\mathbf{M}_\beta \in \mathbb{R}^{V\times 3}$. In addition to the 3D joint locations, hand shape such as finger thickness is also different for each person. To cope with the shape difference, we build two fully-connected layers and estimate $\Delta\mathbf{M}_\beta$ from the identity code β. The hidden

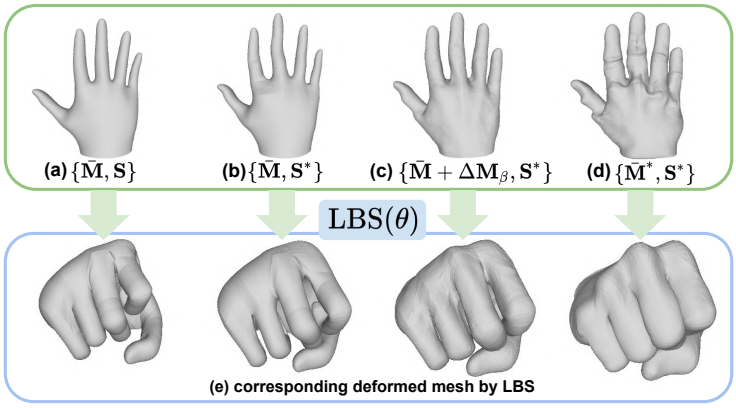

Fig. 4: (a)-(d): Visualized hand model refined by different combinations of correctives. (e): Deformed hand model using LBS.

activation size of the fully-connected layer is 256. The estimated $\Delta\mathbf{M}_\beta$ is added to $\bar{\mathbf{M}}$. Figure 4 (c) shows the effect of $\Delta\mathbf{M}_\beta$.

The last corrective is pose-dependent per-vertex corrective $\Delta\mathbf{M}_\theta \in \mathbb{R}^{V\times 3}$. When making a pose (*i.e.*, θ varies), local deformation of hand geometry such as skin bulging and crease appearing/disappearing also occurs. To recover such phenomena, we build two fully-connected layers to estimate $\Delta\mathbf{M}_\theta$ from the hand pose vector θ. The hidden activation size of the fully-connected layer is 256. The estimated $\Delta\mathbf{M}_\theta$ is added to $\bar{\mathbf{M}}$. For stable training, we do not back-propagate gradient from $\Delta\mathbf{M}_\theta$ through θ. Figure 4 (d) shows the effect of $\Delta\mathbf{M}_\theta$.

The final refined hand model $\mathcal{M}^*$ is obtained as follows:

$$\bar{\mathbf{M}}^* = \bar{\mathbf{M}} + \Delta\mathbf{M}_\theta + \Delta\mathbf{M}_\beta,\ \mathbf{S}^* = \mathbf{S} + \Delta\mathbf{S}_\beta,$$
$$\mathcal{M}^* = \{\bar{\mathbf{M}}^*, \mathbf{S}^*\}.$$

5.2 Hand model deformation

We first perform 3D rigid alignment from the hand model space to the dataset space for the global alignment using the wrist and finger root positions. Then, we use the LBS algorithm to holistically deform our hand model. LBS is a widely used algorithm to deform a mesh according to linear combinations of joint rigid transformation [21, 28]. Specifically, each vertex $\mathbf{m}_v$ of a deformed hand mesh $\mathbf{M} \in \mathbb{R}^{V\times 3}$ is obtained as follows:

$$\begin{aligned}\mathbf{m}_v =& (\mathbf{I}_3, \mathbf{0}) \cdot \sum_{j=1}^{J} w_{v,j} \mathbf{T}_j(\theta, \mathbf{S}^*; \mathcal{H}) \begin{pmatrix} \bar{\mathbf{m}}_v^* \\ 1 \end{pmatrix} \\ =& \text{LBS}(\theta, \mathbf{S}^*, \bar{\mathbf{m}}_v^*), v = 1, \ldots, V,\end{aligned} \tag{1}$$

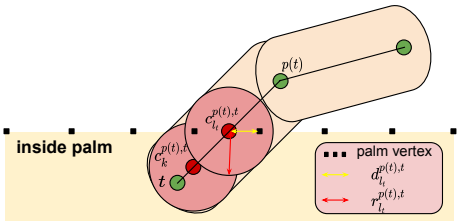

Fig. 5: Visualized example of penetration between a finger and palm.

where $\mathbf{T}_j(\theta, \mathbf{S}^*; \mathcal{H}) \in \mathrm{SE}(3)$ denotes transformation matrix for joint j. It encodes the rotation and translation from the zero pose to the target pose, constructed by traversing the hierarchy $\mathcal{H}$ from the root to j. $w_{v,j}$ and $\bar{\mathbf{m}}_v^*$ denote jth joint of vth vertex skinning weight from $\mathbf{W}$ and vth vertex coordinate from $\bar{\mathbf{M}}^*$, respectively. The visualization of a deformed mesh is shown in Figure 4 (e).

6 Training DeepHandMesh

We use four loss functions to train DeepHandMesh. The **Pose loss** and **Depth map loss** are responsible for the weak supervision. The **Penetration loss** helps to reproduce physical interaction between hand parts and the **Laplacian loss** acts as a regularizer to make output hand meshes smooth.

Pose loss. We perform forward kinematics from the estimated hand pose vector θ and refined skeleton $\mathbf{S}^*$ to get the 3D coordinates of the hand joints $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_J]^T \in \mathbb{R}^{J\times 3}$. We minimize $L1$ distance between the estimated and the groundtruth coordinates. The pose loss is defined as follows: $L_{\text{pose}} = \frac{1}{J}\sum_{j=1}^{J} ||\mathbf{p}_j - \mathbf{p}_j^*||_1$, where $*$ indicates the groundtruth.

Depth map loss. We render 2D depth maps $\mathcal{D} = (\mathbf{D}_1, \dots, \mathbf{D}_{C_{\text{out}}})$ of $\mathbf{M}$ from randomly selected C_{out} target views, and minimize Smooth_{L1} distance [8] between the rendered and the groundtruth depth maps following Ge et al. [7]. To make the depth map loss differentiable, we use Neural Renderer [14]. The depth map loss is defined as follows: $L_{\text{depth}} = \frac{1}{C_{\text{out}}}\sum_{c=1}^{C_{\text{out}}} \delta_c \left(\text{Smooth}_{L1}(\mathbf{D}_c, \mathbf{D}_\mathbf{c}^*)\right)$, where $*$ indicates the groundtruth. δ_c is a binary map whose pixel value of each grid is one if it is foreground (*i.e.*, a depth value is defined in $\mathbf{D}_c$ and $\mathbf{D}_c^*$), and zero otherwise.

Penetration loss. To penalize penetration between hand parts, we introduce two penetration avoidance regularizers. We consider the fingers as rigid hand parts and the palm as a non-rigid hand part. The regularizers are designed for each of the rigid and non-rigid parts.

For the rigid parts (*i.e.*, fingers), we use a regularizer similar to that in Wan et al. [36], which represents each rigid part with a combination of spheres. Specifically, we compute a pair of the center and radius of spheres $\{s_k^{p(j),j} = (c_k^{p(j),j}, r_k^{p(j),j})\}_{k=1}^{K}$ between joint j and its parent joint $p(j)$, where $K = 10$

denotes the number of spheres between the adjacent joints. The center $c_k^{p(j),j}$ is computed by linearly interpolating $\bar{\mathbf{p}}_{p(j)}$ and $\bar{\mathbf{p}}_j$, where $c_1^{p(j),j} = \bar{\mathbf{p}}_{p(j)}$ and $c_K^{p(j),j} = \bar{\mathbf{p}}_j$. $\bar{\mathbf{p}}_j$ denotes the 3D coordinate of hand joint j obtained from forward kinematics using $\theta = \mathbf{0}$ and $\mathbf{S}^*$. Each radius $r_k^{p(j),j}$ is obtained by calculating the distance between $c_k^{p(j),j}$ and the closest vertex in $\text{LBS}(\mathbf{0}, \mathbf{S}^*, \bar{\mathbf{M}}^*)$. Given these spheres, the penetration avoidance term between the rigid hand parts $L_{\text{penet}}^{\text{r}}$ is defined as follows:

$$L_{\text{penet}}^{\text{r}} = \sum_{\substack{k,k\prime \\ j\neq j\prime, p(j\prime) \\ j\prime\neq p(j)}} \max(r_k^{p(j),j} + r_{k\prime}^{p(j\prime),j\prime} - ||c_k^{p(j),j} - c_{k\prime}^{p(j\prime),j\prime}||_2, 0), \tag{2}$$

which indicates that the distances of any pairs of the spheres except the ones associated with adjacent joints are enforced to be greater than the sum of the radii of the paired spheres. This prevents overlap between the spheres, thus avoiding penetration between the rigid parts.

However, $L_{\text{penet}}^{\text{r}}$ does not help prevent penetration at the non-rigid hand part (*i.e.*, the palm). The underlying assumption of $L_{\text{penet}}^{\text{r}}$ is that surface geometry can be approximated by many spheres. While this assumption holds for the fingers due to the cylindrical shape, it does not often hold for the palm, *i.e.*, the spheres along the joints in the palm cannot approximate the palm surface particularly when pose-dependent corrective replicating skin bulging is applied. Additionally, $L_{\text{penet}}^{\text{r}}$ does not produce surface deformation, *e.g.*, finger-palm collision often makes large deformation to the palm surface. $L_{\text{penet}}^{\text{r}}$ does not help replicate such deformation.

To address those limitations, we propose a new penetration avoidance term $L_{\text{penet}}^{\text{nr}}$ for the non-rigid hand part. For this, we only consider penetration between fingertips and palm as illustrated in Figure 5. Among $\mathbf{M}$, vertices whose most dominant joint in the skinning weight $\mathbf{W}$ is the palm are considered as ones for the palm $\mathbf{M}_\gamma$. Then, the distance between $c_k^{p(t),t}$ and $\mathbf{M}_\gamma$ is calculated, where t is one of fingertip joints. Among the distances, the shortest one is denoted as $d_k^{p(t),t}$. If there exists l_t where $d_{l_t}^{p(t),t}$ is smaller than $r_{l_t}^{p(t),t}$, we consider that $c_{l_t}^{p(t),t}$ penetrates $\mathbf{M}_\gamma$. If there are more than one l_t, we use the one closest to the $p(t)$, which is considered as a starting point of penetration. Based on human hand anatomical property, we can conclude that the spheres from l_t to the fingertip $\{s_k^{p(t),t}\}_{k=l_t}^K$ are penetrating $\mathbf{M}_\gamma$. Then, we enforce $\{d_k^{p(t),t}\}_{k=l_t}^K$ to be the same as $\{r_k^{p(t),t}\}_{k=l_t}^K$. The penetration avoidance term for the non-rigid hand part $L_{\text{penet}}^{\text{nr}}$ is defined as follows:

$$L_{\text{penet}}^{\text{nr}} = \sum_t g(t), \tag{3}$$

$$\text{where } g(t) = \begin{cases} \sum_{k=l_t}^K |d_k^{p(t),t} - r_k^{p(t),t}|, & \text{if } l_t \text{ exists} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The final penetration avoidance loss function is defined as follows: $L_{\text{penet}} = L_{\text{penet}}^{\text{r}} + \lambda_{\text{nr}} L_{\text{penet}}^{\text{nr}}$, where $\lambda_{\text{nr}} = 5$.

Laplacian loss. To preserve local geometric structure of the deformed mesh based on the mesh topology, we add a Laplacian regualarizer [19] as follows: $L_{\text{lap}} = \frac{1}{V}\sum_{v=1}^{V}\left(\mathbf{m}_v - \frac{1}{||\mathcal{N}(v)||}\sum_{v'\in\mathcal{N}(v)}\mathbf{m}_{v'}\right)$, where $\mathcal{N}(v)$ denotes neighbor vertices of $\mathbf{m}_v$.

Our DeepHandMesh is trained in an end-to-end manner. Note that although our DeepHandMesh is trained without per-vertex mesh supervision, it can be trained with a single regularizer L_{lap}. The total loss function L is defined as follows: $L = L_{\text{pose}} + L_{\text{depth}} + L_{\text{penet}} + \lambda_{\text{lap}}L_{\text{lap}}$, where $\lambda_{\text{lap}} = 5$.

7 Implementation details

PyTorch [26] is used for implementation. The ResNet in the encoder is initialized with the publicly released weights pre-trained on the ImageNet dataset [29], and the weights of the remaining part are initialized by Gaussian distribution with zero mean and $\sigma = 0.01$. The weights are updated by the Adam optimizer [16] with a mini-batch size of 32. The number of rendering views is $C_{\text{out}} = 6$. We use 256×256 as the size of $\mathbf{I}$ and depth maps of $\mathcal{D}$. We observed that changing C_{out} and resolution of $\mathbf{I}$ and depth maps of $\mathcal{D}$ does not affect much the quality of the resulting mesh. The number of vertices in our hand model is 12,553. We train our DeepHandMesh for 35 epochs with a learning rate of 10^{-4}. The learning rate is reduced by a factor of 10 at the 30^{th} and 32^{nd} epochs. We used four NVIDIA Titan V GPUs for training, which took 9 hours. Both the encoder and decoder of our DeepHandMesh run at 100 fps, yielding real-time performance (50 fps).

8 Experiment

8.1 Dataset

We used the same data capture studio with Moon et al. [23]. The experimental image data was captured by 80 calibrated cameras capable of synchronously capturing images with 4096 × 2668 pixels at 30 frames per second. All cameras lie on the front, side, and top hemisphere of the hand and are placed at a distance of about one meter from it. During capture, each subject was instructed to make a pre-defined set of 40 hand motions and 15 conversational gestures. We pre-processed the raw video data by performing multi-view 3D hand pose estimation [18] and multi-view 3D reconstruction [5]. We split our dataset into training and testing sets. The training set contains 404K images per subject with the 40 pre-defined hand poses, and the test set contains 80K images per subject with the 15 conversational gestures. There are four subjects (one female and three males), and we show more detailed description and various examples of our dataset in the supplementary material.

8.2 Ablation study

Effect of each loss function. To investigate the effect of each loss function, we visualize test results from models trained with different combinations of loss

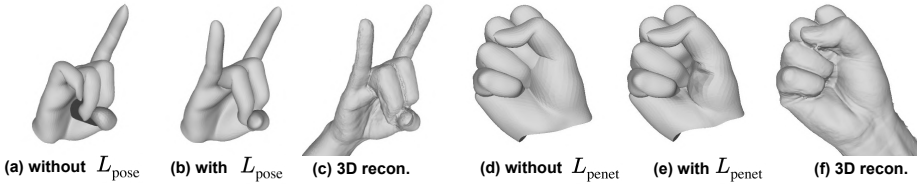

Fig. 6: (a)-(c): Deformed hand mesh trained without and with L_{pose}, and corresponding 3D reconstruction [5]. (d)-(f): Deformed hand mesh trained without and with L_{penet}, and corresponding 3D reconstruction [5].

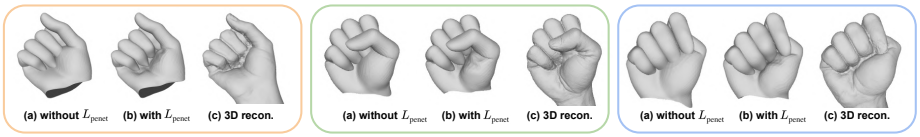

Fig. 7: Deformed hand mesh trained without and with L_{penet}, and corresponding 3D reconstruction [5].

functions in Figure 6. In the figure, (a), (b), (d), and (e) are the results of our DeepHandMesh, and (c) and (f) are the results of 3D reconstruction [5], respectively.

The model trained without L_{pose} (a) gives wrong joint locations. Also, there are severe artifacts at occluded hand regions (*e.g.*, the black area on the palm region) because of skin penetration. This is because L_{depth} cannot back-propagate gradients through occluded areas. In contrast, L_{pose} can give gradients at the invisible regions, which makes more stable and accurate results, as shown in (b). The model trained without L_{penet} (d) cannot prevent penetration between fingers and palm. However, L_{penet} penalizes this, and the fingertip locations are placed more plausibly, and the palm vertices are deformed according to the physical interaction between the fingers and palm, as shown in (e). Figure 7 additionally shows the effectiveness of the proposed L_{penet}.

Effect of identity-dependent correctives. To demonstrate the effectiveness of our identity-dependent corrective (*i.e.*, $\Delta\mathbf{S}_\beta$ and $\Delta\mathbf{M}_\beta$), we visualize how our DeepHandMesh handles different identities in Figure 8. The figures are drawn by setting $\theta = \mathbf{0}$ to normalize hand pose. As the figures show, our identity-dependent corrective successfully personalizes the initial hand model to each subject by adjusting the hand bone lengths and skin.

Effect of pose-dependent corrective. To demonstrate the effectiveness of our pose-dependent per-vertex corrective $\Delta\mathbf{M}_\theta$, we visualize the hand meshes of different poses in Figure 9. All the hand meshes are from the same subject to normalize identity. For each hand pose, (a) shows the hand model after model refinement with zero pose. (b) shows deformed (a) using LBS, and (c) shows

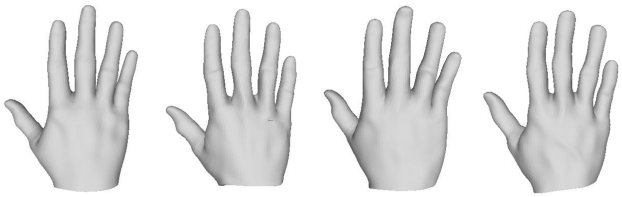

Fig. 8: Visualized hand models of zero pose from different subjects.

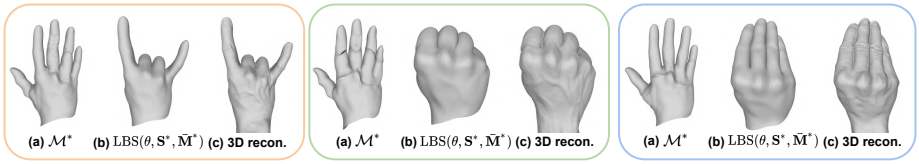

Fig. 9: (a) Refined hand model, (b) deformed hand mesh, and (c) 3D reconstruction [5] from various hand poses of a one subject.

3D reconstruction meshes. As the figure shows, our pose-dependent correctives successfully recover details according to the poses. Note that in (b), we approximately reproduced local deformation based on the blood vessels.

8.3 Comparison with state-of-the-art methods

We compare our DeepHandMesh with widely used hand model MANO [28] on our dataset. For comparison, we train a model whose encoder is the same one as ours, and decoder is the pre-trained MANO model. The pre-trained MANO model is fixed during the training, and we use the same loss functions as ours. We pre-defined identity code β for each subject and estimate the shape vector of MANO from the code using two fully-connected layers to compare both models in the personalized environment. Figure 10 shows the proposed DeepHandMesh provides significantly more realistic hand mesh from various hand poses and identities. In the last row, MANO suffers from the unrealistic physical interaction between hand parts such as finger penetration and flat palm skin. In contrast, our DeepHandMesh does not suffer from finger penetration and can replicate physical interaction between finger and palm skin. Table 1 shows the 3D joint coordinate distance error and mesh vertex error from the closest point on the 3D reconstruction meshes for unseen hand poses, indicating that our DeepHandMesh outperforms MANO on the unseen hand pose images. For more comparisons, we experimented with lower-resolution hand mesh in the supplementary material.

We found that comparisons between DeepHandMesh and MANO with publicly available 3D hand datasets [41, 43] were difficult because DeepHandMesh assumes a personalized environment (*i.e.*, assumes the same subject in training

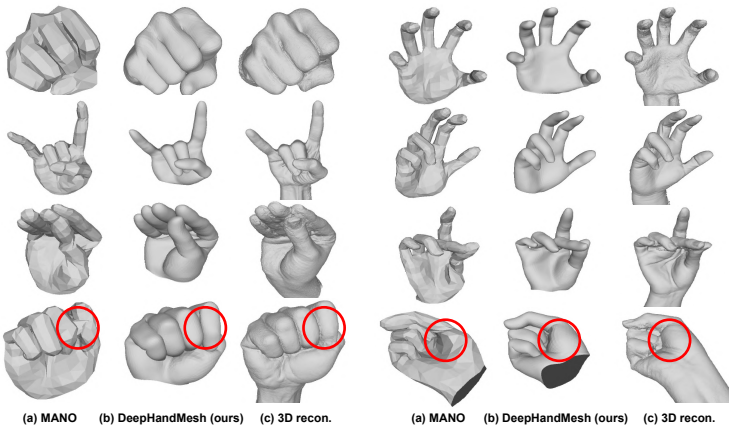

Fig. 10: Estimated hand mesh comparison from various hand poses and subjects with the state-of-the-art method. The red circles in the last row show physical interaction between hand parts.

methods	$\mathbf{P}_{\text{err}}$ (mm)	$\mathbf{M}_{\text{err}}$ (mm)
MANO	13.81	8.93
DeepHandMesh (Ours)	**9.86**	**6.55**

Table 1: 3D joint distance error $\mathbf{P}_{\text{err}}$ and mesh vertex error $\mathbf{M}_{\text{err}}$ comparison between MANO and DeepHandMesh on test set consists of unseen hand poses.

and testing stages). However, we believe the qualitative and quantitative comparisons in Figure 10 and Table 1 still show the superiority of the proposed DeepHandMesh.

8.4 3D hand mesh estimation from general images

To demonstrate a use case of DeepHandMesh for general images, we developed a model-based 3D hand mesh estimation system based on DeepHandMesh. Figure 11 shows that our model-based 3D hand mesh estimation system generates realistic hand meshes without mesh supervision from the test set of the RHD [43]. For this, we first pre-trained DeepHandMesh, and replaced its encoder with a randomly initialized one that has exactly the same architecture with our encoder, as illustrated in Figure 2. We trained the new encoder on the training set of RHD, by minimizing L_{pose}. The RHD dataset contains 44K images synthesized by animating the 3D human models. During the training, the decoder is fixed, which is a similar training strategy with that of MANO-based 3D hand mesh estimation methods [1, 3]. As our DeepHandMesh assumes a personalized environment, we used a groundtruth bone length to adjust a bone length of the output 3D joint coordinates. The inputs of the decoder are joint rotations and

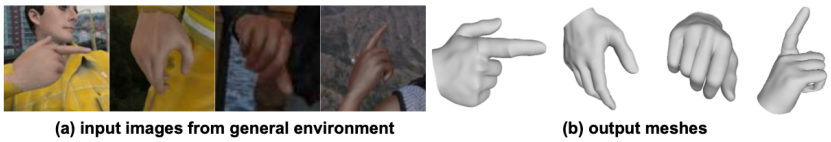

Fig. 11: 3D hand mesh estimation results from general images.

identity code without any image appearance information like MANO; therefore, the decoder can easily generalize to general images, although it is trained on the data captured from the controlled environment.

9 Discussion

Our DeepHandMesh assumes a personalized environment. Future work should consider cross-identity hand mesh modeling by estimating the Gaussian identity code. However, training cross-identity hand mesh model in a weakly-supervised way is very hard. As MANO is trained in a fully-supervised way, they could perform principal component analysis (PCA) on the groundtruth hand meshes in zero-pose and model the identity as coefficients of the principal components. On the other hand, there is no groundtruth mesh under the weakly-supervised setting, therefore performing PCA on meshes is not possible. Generative models (*e.g.*, VAE) can be designed to learn a latent space of identities from registered meshes like [13]; however, training a generative model in a weakly supervised way without registered meshes also remains challenging. We believe the extension of DeepHandMesh to handle cross-identity in a weakly-supervised setting could be an interesting future direction.

10 Conclusion

We presented a novel and powerful weakly-supervised deep encoder-decoder framework, DeepHandMesh, for high-fidelity hand mesh modeling. In contrast to the previous hand models [17, 28], DeepHandMesh is trained in a weakly-supervised setting; therefore, it does not require groundtruth hand mesh. Our model successfully generates more realistic hand mesh compared with the previous fully-supervised hand models. The newly introduced penetration avoidance loss makes the result even more realistic by replicating physical interactions between hand parts.

Acknowledgments

This work was partially supported by the Next-Generation Information Computing Development Program (NRF-2017M3C4A7069369) and the Visual Turing Test project (IITP-2017-0-01780) funded by the Ministry of Science and ICT of Korea.

Supplementary Material of "DeepHandMesh: A Weakly-Supervised Deep Encoder-Decoder Framework for High-Fidelity Hand Mesh Modeling"

In this supplementary material, we present more experimental results that could not be included in the main manuscript due to the lack of space.

11 Texture loss

Although the overall shape of the hand mesh is close to the groundtruth, some vertices can move inconsistently across time steps. To prevent this, we employ texture consistency loss, similar to [37,39]. Specifically, we first train the DeepHandMesh and obtain a hand mesh of a neutral pose, which is considered as the easiest pose to estimate. Then, the mesh is used to unwrap neutral pose RGB images from all views of our dataset to a 1024×1024 UV texture $\mathbf{T}^*$ using Poisson reconstruction [15]. We use $\mathbf{T}^*$ as a groundtruth texture and force texture $\mathbf{T}$ of each iteration to be the same with $\mathbf{T}^*$. $\mathbf{T}$ is obtained by unwrapping corresponding RGB images from randomly selected C_{out} views using mesh output of current iteration in a differentiable way [20,37]. The resolution of $\mathbf{T}$ is 256×256, and we resized $\mathbf{T}^*$ to the same resolution of $\mathbf{T}$. To normalize illumination, we perform normalized cross-correlation (NCC) on each 8×8 patch of $\mathbf{T}$ and $\mathbf{T}^*$ after applying the average blur. The loss function L_{tex} is defined as follows:

$$L_{\text{tex}} = \delta||(\text{NCC}(\mathbf{T}) - \text{NCC}(\mathbf{T}^*))||_1, \tag{5}$$

where δ is a binary tensor whose value is one if the corresponding UV coordinate has visible RGB value. This loss function is applied to fine-tune trained DeepHandMesh 15 epochs. The total loss function in fine-tuning stage is $L + L_{\text{tex}}$. During fine-tuning, we used the same learning rate 10^{-4}, and it is reduced by a factor of 10 at the 10th and 12th epochs. λ_{lap} is set to 1.

Figure 12 shows standard deviation σ of each pixel value on the UV space after fine-tuning without L_{tex} (a) and with (b). (c) shows σ difference between (a) and (b), which is defined as (b) subtracted by (a) (*i.e.*, blue colors in (c) indicate σ decreased after fine-tuning). As the figures show, L_{tex} helps to decrease σ, but not at a noticeable amount. We guess that this is because (a) already shows low standard deviation.

12 Effect of the skeleton corrective

To demonstrate the effectiveness of the identity-dependent skeleton corrective $\Delta\mathbf{S}_\beta$, we compare 3D joint distance error P_{err} in Table 2. The error is defined as a Euclidean distance between $\mathbf{P}$ and $\mathbf{P}^*$, where $*$ indicates groundtruth. The table shows our skeleton corrective refines the skeleton of the initial hand model successfully.

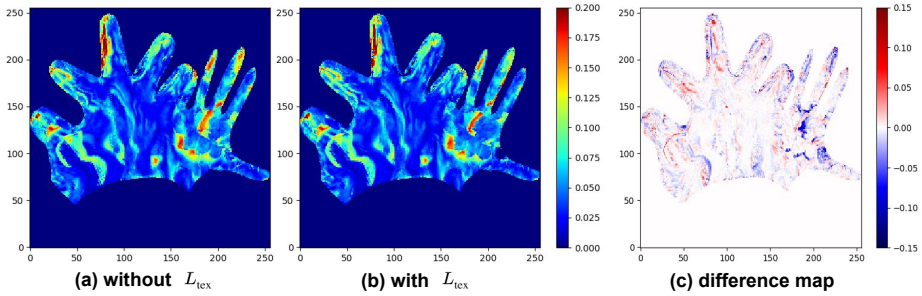

Fig. 12: Visualization of effect of L_{tex}.

Settings	P_{err} (mm)
Without $\Delta\mathbf{S}_\beta$	4.82
Ours (full)	**2.38**

Table 2: 3D joint distance error P_{err} (mm) comparison between with and without our identity-dependent skeleton corrective $\Delta\mathbf{S}_\beta$.

13 Skinning weight corrective

To refine pre-defined skinning weight $\mathbf{W}$, we estimate identity-dependent skinning weight corrective $\Delta\mathbf{W}_\beta \in \mathbb{R}^{V\times J}$ from the identity vector β using two fully-connected layers. The refined skinning weight $\mathbf{W}^* \in \mathbb{R}^{V\times J}$ is obtained as follows:

$$\mathbf{w}^*_{v,j} = \frac{\max(\mathbf{w}_{v,j} + \Delta\mathbf{w}_{v,j}, 0)}{\sum_{j=1}^{J} \max(\mathbf{w}_{v,j} + \Delta\mathbf{w}_{v,j}, 0)}, v = 1, \dots, V, j = 1, \dots, J,$$

where $\mathbf{w}^*_{v,j}$, $\mathbf{w}_{v,j}$, and $\Delta\mathbf{w}_{v,j}$ denote refined skinning weight, initial skinning weight, and skinning weight corrective of jth joint of vth vertex from $\mathbf{W}^*$, $\mathbf{W}$, and $\Delta\mathbf{W}_\beta$, respectively. We clamp the refined skinning weight to be positive value and normalize it to make the summation 1. To encourage locality like Loper et al. [21], $\Delta\mathbf{w}_{v,j}$ is estimated only when $\mathbf{w}_{v,j} \neq 0$. Otherwise, it is set to zero.

We trained the DeepHandMesh with an additional $\Delta\mathbf{W}_\beta$. Figure 13 shows color-coded skinning weight and deformed hand mesh. As the figure shows, the skinning weight of mainly finger root parts changed, and this change results in different skin deformation around the finger root parts. However, as there is no groundtruth mesh, it is hard to tell which hand mesh is more realistic clearly. We believe this skinning weight corrective $\Delta\mathbf{W}_\beta$ can be helpful when initial skinning weight $\mathbf{W}$ is bad and a bottleneck of better performance.

14 Dataset description

We provide detailed descriptions and visualizations of the sequences in our newly constructed dataset. The pre-defined hand poses include various sign languages

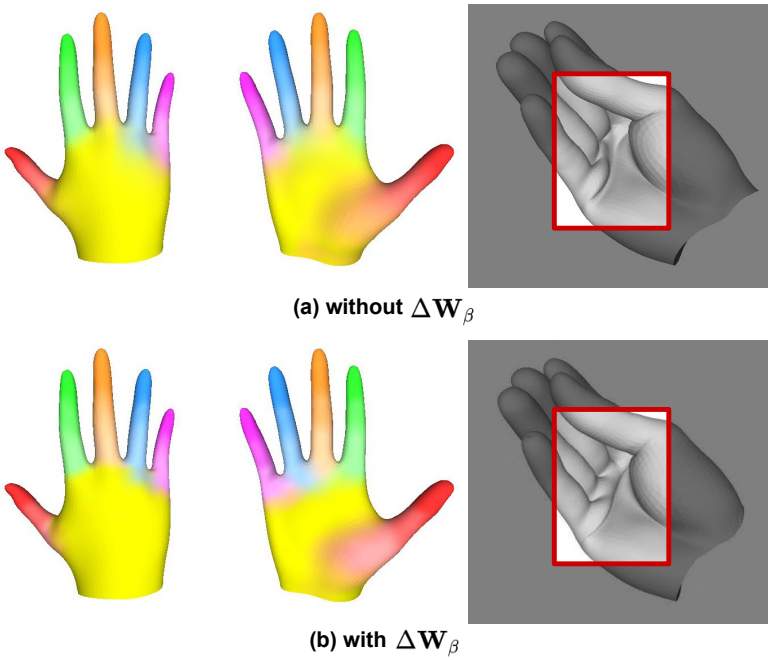

Fig. 13: Color-coded skinning weight (left and middle) and deformed hand mesh (right) comparison between without and with skinning weight corrective $\Delta \mathbf{W}_\beta$.

that are frequently used in daily life and extreme poses where each finger assumes a maximally bent or extended. When capturing the conversational gestures, subjects are instructed with minimal instructions, for example, waving their hands as if telling someone to come over. The hand poses in our dataset are carefully chosen to sample a variety of poses and conversational gestures while being easy to follow by capture participants.

The hand pose estimator was trained on our held-out human annotation dataset, which includes the 3D rotation center coordinates of hand joints from 6K frames. The predicted 2D hand joint locations of each view were triangulated with RANSAC to robustly obtain the groundtruth 3D hand joint coordinates. The hand pose estimator used to obtain groundtruth 3D hand joint coordinates achieves *2.78 mm error* on our held-out human-annotated test set, which is quite low. From the 3D reconstruction, we rendered groundtruth depth maps for all camera views.

Training set. Figure 14, 15, and 16 show the pre-defined hand poses in training set. Belows are detailed descriptions of each sequence.

- neutral relaxed: the neutral hand pose. Hands in front of the chest, fingers do not touch, and palms face the side.
- neutral rigid: the neutral hand pose with maximally extended fingers, muscles tense.

- good luck: hand sign language with crossed index and middle fingers.
- fake gun: hand gesture mimicking the gun.
- star trek: hand gesture popularized by the television series Star Trek.
- star trek extended thumb: "star trek" with extended thumb.
- thumb up relaxed: hand sign language that means "good", hand muscles relaxed.
- thumb up normal: "thumb up", hand muscles average tenseness.
- thumb up rigid: "thumb up", hand muscles very tense.
- thumb tuck normal: similar to fist, but the thumb is hidden by other fingers.
- thumb tuck rigid: "thumb tuck", hand muscles very tense.
- aokay: hand sign language that means "okay", where palm faces the side.
- aokay upright: "aokay" where palm faces the front.
- surfer: the SHAKA sign.
- rocker: hand gesture that represents rock and roll, where palm faces the side.
- rocker front: the "rocker" where palm faces the front.
- rocker back: the "rocker" where palm faces the back.
- fist: fist hand pose.
- fist rigid: fist with very tense hand muscles.
- alligator closed: hand gesture mimicking the alligator with a closed mouth.
- one count: hand sign language that represents "one."
- two count: hand sign language that represents "two."
- three count: hand sign language that represents "three."
- four count: hand sign language that represents "four."
- five count: hand sign language that represents "five."
- indextip: thumb and index fingertip are touching.
- middletip: thumb and middle fingertip are touching.
- ringtip: thumb and ring fingertip are touching.
- pinkytip: thumb and pinky fingertip are touching.
- palm up: has palm facing up.
- finger spread relaxed: spread all fingers, hand muscles relaxed.
- finger spread normal: spread all fingers, hand muscles average tenseness.
- finger spread rigid: spread all fingers, hand muscles very tense.
- capisce: hand sign language that represents "got it" in Italian.
- claws: hand pose mimicking claws of animals.
- peacock: hand pose mimicking peacock.
- cup: hand pose mimicking a cup.
- shakespeareyorick: hand pose from Yorick from Shakespeare's play Hamlet.
- dinosaur: hand pose mimicking a dinosaur.
- middle finger: hand sign language that has an offensive meaning.

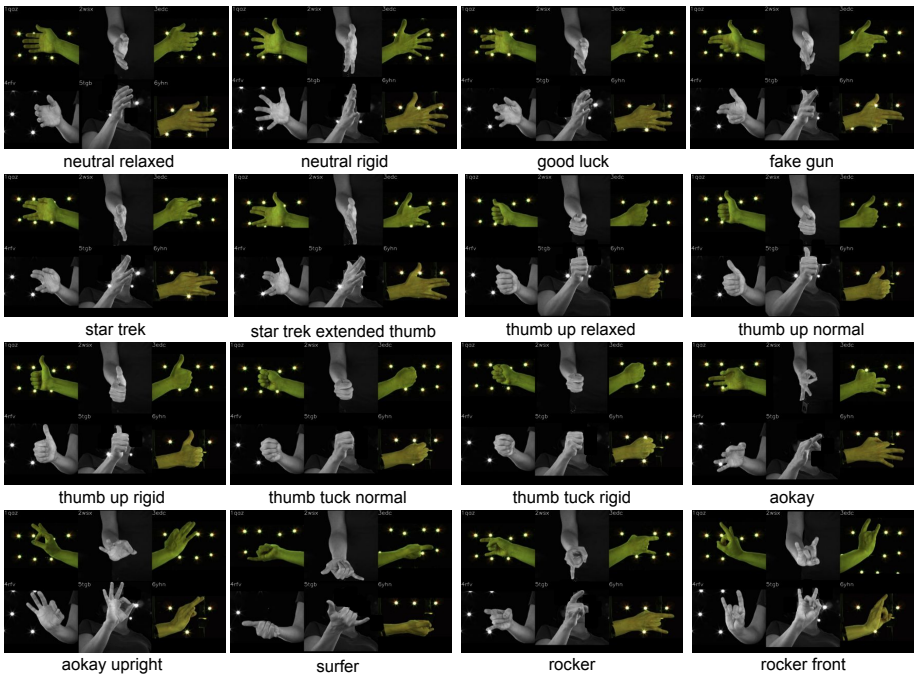

Fig. 14: Visualization of the sequences in the training set.

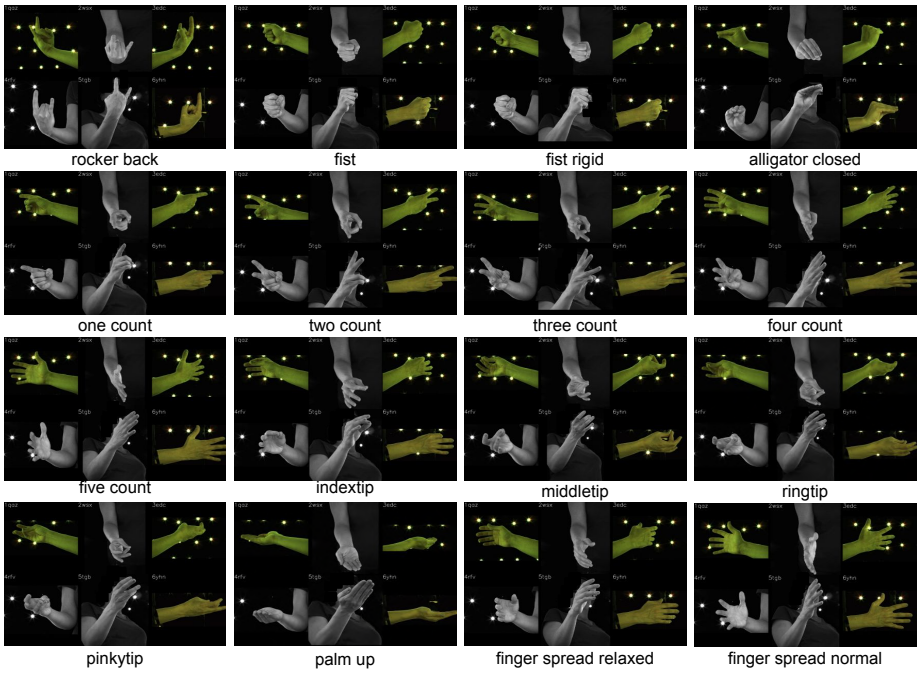

Fig. 15: Visualization of the sequences in the training set.

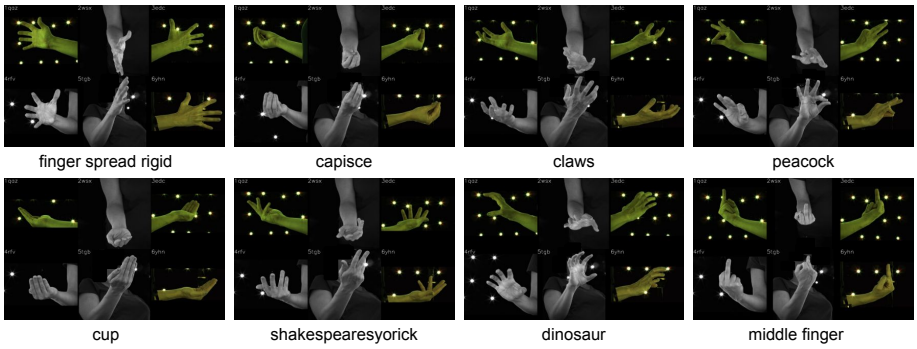

Fig. 16: Visualization of the sequences in the training set.

Testing set. Figure 17 shows the conversational gestures in testing set. Belows are detailed descriptions of each sequence.

- five count: count from one to five.
- five countdown: count from five to one.
- fingertip touch: thumb touch each fingertip.
- relaxed wave: wrist relaxed, fingertips facing down and relaxed, wave.
- fist wave: rotate wrist while hand in a fist shape.
- prom wave: wave with fingers together.
- palm down wave: wave hand with the palm facing down.
- index finger wave: hand gesture that represents "no" sign.
- palmer wave: palm down, scoop towards you, like petting an animal.
- snap: snap middle finger and thumb.
- finger wave: palm down, move fingers like playing the piano.
- finger walk: mimicking a walking person by index and middle finger.
- cash money: rub thumb on the index and middle fingertips.
- snap all: snap each finger on the thumb.

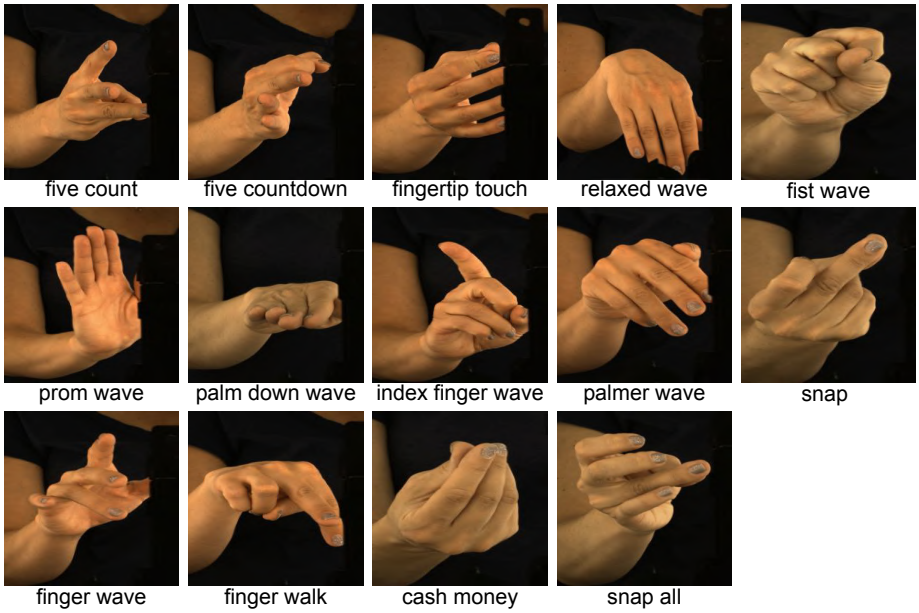

Fig. 17: Visualization of the sequences in the testing set.

Figure 18 shows rendering of our constructed multi-view studio for the data capture.

Fig. 18: Rendering of our constructed multi-view studio.

15 Comparison with state-of-the-art methods

Comparison with MANO under the similar mesh resolution. We train and test our DeepHandMesh with a hand model the resolution of which is similar to that of MANO, and compare its qualitative results with those from MANO in Figure 19. Our low-resolution DeepHandMesh uses a hand mesh model with 792 vertices, while the MANO is based on a hand mesh model with 778 vertices. The figure shows that our DeepHandMesh provides a more realistic hand mesh compared with MANO under the similar resolution of the hand mesh model. Note that our DeepHandMesh is trained in a weakly-supervised way without per-vertex loss function, while MANO is based on fully-supervised training with per-vertex loss. When we train low resolution version of the DeepHandMesh, $L2$ norm regularizers are used for the correctives (*i.e.*, $\Delta\mathbf{S}_\beta$, $\Delta\mathbf{M}_\beta$, and $\Delta\mathbf{M}_\theta$).

Comparison with MANO on the dataset of MANO. We tried to train and test our DeepHandMesh on MANO dataset [28]. However, we observed that there are only 50 registrations available for each subject, which are not large enough to train DeepHandMesh. Also, some of the 3D scans include an object grasped by a hand. This makes training our system on the MANO dataset hard because rendered groundtruth depth maps $\mathcal{D}^*$ include those objects. Although we tried to use the registered meshes that do not include the objects for depth map rendering, we noticed that the rendered depth map lost high-frequency information in the original 3D scans because of the low-resolution mesh in MANO, which makes the depth maps hard to be used as groundtruth depth maps $\mathcal{D}^*$.

Comparison with Kulon et al. [17]. We also tried to compare our DeepHandMesh with Kulon et al. [17]. They use mesh supervision when they train their high-resolution hand model (*i.e*, 7,907 vertices). As there is no mesh groundtruth in our dataset, we trained their model with the same loss functions as ours (*i.e.*, **Pose loss** and **Depth map loss**). We observed that without per-vertex mesh supervision, their model provides severely distorted hand mesh. We could not train our DeepHandMesh on the Panoptic dome dataset [31] that Kulon et al. [17] used because high-quality multi-view depth maps are not available in that dataset. Instead, we compare hand mesh output of the same hand pose from our DeepHandMesh and Kulon et al. [17] trained on our dataset and the Panoptic dome dataset [31], respectively. Although this is not a perfectly fair comparison, we think that this can roughly show how the final outputs of each method are different. Figure 20 shows our DeepHandMesh provides significantly more realistic hand mesh than Kulon et al. [17].

Comparison on publicly available datasets. As our DeepHandMesh is a personalized system, it is hard to compare with other hand models (*i.e.*, MANO [28] and Kulon et al. [17]) that support cross-identity on publicly available datasets. Instead, we tried to best to provide comparisons between them on our dataset.

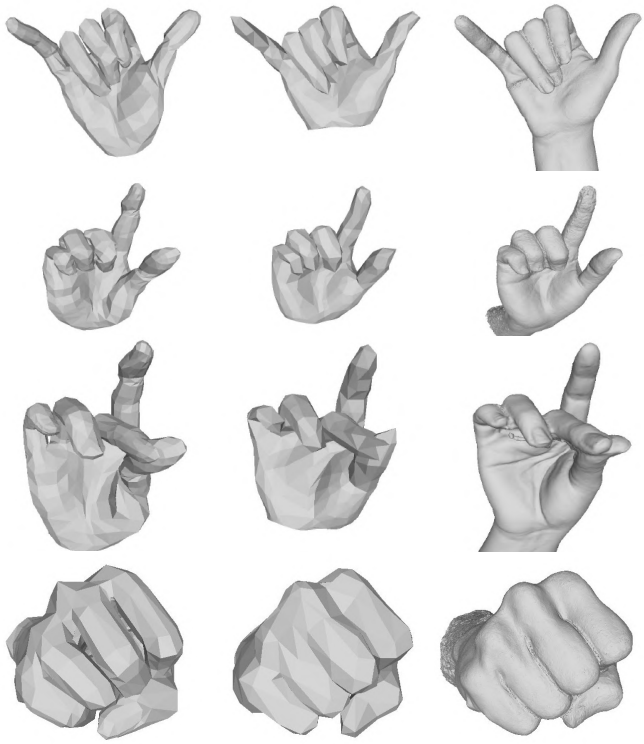

Fig. 19: Estimated hand mesh comparison with MANO [28] and our DeepHandMesh using the hand model of the similar resolution.

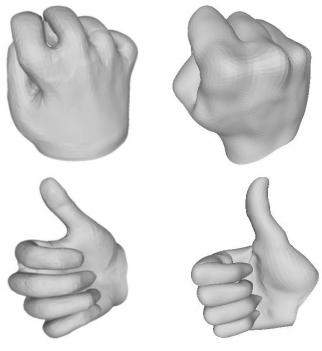

Fig. 20: Estimated hand mesh comparison with Kulon et al. [17] and our DeepHandMesh. The results of Kulon et al. [17] are taken from their paper.

16 Qualitative rendered results

We provide rendered result using texture obtained from Section 11 in Figure 21.

Fig. 21: Comparison between our rendered image and captured image from cameras.

References

1. Baek, S., In Kim, K., Kim, T.K.: Pushing the envelope for RGB-based dense 3D hand pose estimation via neural rendering. In: CVPR (2019)
2. Bogo, F., Romero, J., Loper, M., Black, M.J.: FAUST: Dataset and evaluation for 3D mesh registration. In: CVPR (2014)
3. Boukhayma, A., de Bem, R., Torr, P.H.: 3D hand shape and pose from images in the wild. In: CVPR (2019)
4. Cai, Y., Ge, L., Cai, J., Yuan, J.: Weakly-supervised 3D hand pose estimation from monocular RGB images. In: ECCV (2018)
5. Galliani, S., Lasinger, K., Schindler, K.: Massively parallel multiview stereopsis by surface normal diffusion. In: ICCV (2015)
6. Ge, L., Liang, H., Yuan, J., Thalmann, D.: Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs. In: CVPR (2016)
7. Ge, L., Ren, Z., Li, Y., Xue, Z., Wang, Y., Cai, J., Yuan, J.: 3D hand shape and pose estimation from a single RGB image. In: CVPR (2019)
8. Girshick, R.: Fast r-cnn. In: ICCV (2015)
9. Hasson, Y., Varol, G., Tzionas, D., Kalevatykh, I., Black, M.J., Laptev, I., Schmid, C.: Learning joint reconstruction of hands and manipulated objects. In: CVPR (2019)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
11. Hirshberg, D.A., Loper, M., Rachlin, E., Black, M.J.: Coregistration: Simultaneous alignment and modeling of articulated 3D shape. In: ECCV (2012)
12. Iqbal, U., Molchanov, P., Breuel Juergen Gall, T., Kautz, J.: Hand pose estimation via latent 2.5D heatmap regression. In: ECCV (2018)
13. Jiang, Z.H., Wu, Q., Chen, K., Zhang, J.: Disentangled representation learning for 3D face shape. In: CVPR (2019)
14. Kato, H., Ushiku, Y., Harada, T.: Neural 3D mesh renderer. In: CVPR (2018)
15. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. vol. 7 (2006)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. ICLR (2014)
17. Kulon, D., Wang, H., Güler, R.A., Bronstein, M., Zafeiriou, S.: Single image 3D hand reconstruction with mesh convolutions. BMVC (2019)
18. Li, W., Wang, Z., Yin, B., Peng, Q., Du, Y., Xiao, T., Yu, G., Lu, H., Wei, Y., Sun, J.: Rethinking on multi-stage networks for human pose estimation. arXiv preprint arXiv:1901.00148 (2019)
19. Liu, S., Chen, W., Li, T., Li, H.: Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. ICCV (2019)
20. Lombardi, S., Saragih, J., Simon, T., Sheikh, Y.: Deep appearance models for face rendering. ACM TOG (2018)
21. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. ACM TOG (2015)
22. Moon, G., Yong Chang, J., Mu Lee, K.: V2V-PoseNet: Voxel-to-voxel prediction network for accurate 3D hand and human pose estimation from a single depth map. In: CVPR (2018)
23. Moon, G., Yu, S.I., Wen, H., Shiratori, T., Lee, K.M.: InterHand2.6M: A dataset and baseline for 3D interacting hand pose estimation from a single RGB image. In: ECCV (2020)

24. Mueller, F., Bernard, F., Sotnychenko, O., Mehta, D., Sridhar, S., Casas, D., Theobalt, C.: Ganerated hands for real-time 3D hand tracking from monocular RGB. In: CVPR (2018)
25. Panteleris, P., Oikonomidis, I., Argyros, A.: Using a single RGB frame for real time 3D hand pose estimation in the wild. In: WACV (2018)
26. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch (2017)
27. Qian, C., Sun, X., Wei, Y., Tang, X., Sun, J.: Realtime and robust hand tracking from depth. In: CVPR (2014)
28. Romero, J., Tzionas, D., Black, M.J.: Embodied hands: Modeling and capturing hands and bodies together. ACM TOG (2017)
29. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: ImageNet large scale visual recognition challenge. IJCV (2015)
30. Sharp, T., Keskin, C., Robertson, D., Taylor, J., Shotton, J., Kim, D., Rhemann, C., Leichter, I., Vinnikov, A., Wei, Y., et al.: Accurate, robust, and flexible real-time hand tracking. In: ACM Conference on Human Factors in Computing Systems (2015)
31. Simon, T., Joo, H., Matthews, I., Sheikh, Y.: Hand keypoint detection in single images using multiview bootstrapping. In: CVPR (2017)
32. Spurr, A., Song, J., Park, S., Hilliges, O.: Cross-modal deep variational hand pose estimation. In: CVPR (2018)
33. Tagliasacchi, A., Schröder, M., Tkach, A., Bouaziz, S., Botsch, M., Pauly, M.: Robust articulated-ICP for real-time hand tracking. In: Computer Graphics Forum (2015)
34. Tang, D., Taylor, J., Kohli, P., Keskin, C., Kim, T.K., Shotton, J.: Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In: ICCV (2015)
35. Tompson, J., Stein, M., Lecun, Y., Perlin, K.: Real-time continuous pose recovery of human hands using convolutional networks. ACM TOG (2014)
36. Wan, C., Probst, T., Gool, L.V., Yao, A.: Self-supervised 3D hand pose estimation through training by fitting. In: CVPR (2019)
37. Wei, S.E., Saragih, J., Simon, T., Harley, A.W., Lombardi, S., Perdoch, M., Hypes, A., Wang, D., Badino, H., Sheikh, Y.: VR facial animation via multiview image translation. ACM TOG (2019)
38. Yang, L., Yao, A.: Disentangling latent hands for image synthesis and pose estimation. In: CVPR (2019)
39. Yoon, J.S., Shiratori, T., Yu, S.I., Park, H.S.: Self-supervised adaptation of high-fidelity face models for monocular performance tracking. In: CVPR (2019)
40. Yuan, S., Ye, Q., Stenger, B., Jain, S., Kim, T.K.: BigHand2.2M benchmark: Hand pose dataset and state of the art analysis. In: CVPR (2017)
41. Zhang, J., Jiao, J., Chen, M., Qu, L., Xu, X., Yang, Q.: 3D hand pose tracking and estimation using stereo matching. ICIP (2017)
42. Zhou, X., Wan, Q., Zhang, W., Xue, X., Wei, Y.: Model-based deep hand pose estimation. In: IJCAI (2016)
43. Zimmermann, C., Brox, T.: Learning to estimate 3D hand pose from single RGB images. In: ICCV (2017)
44. Zimmermann, C., Ceylan, D., Yang, J., Russell, B., Argus, M., Brox, T.: Freihand: A dataset for markerless capture of hand pose and shape from single RGB images. In: ICCV (2019)