2021 IEEE/CVF International Conference on Computer Vision (ICCV) | 978-1-6654-2812-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCV48922.2021.01149

Towards Accurate Alignment in Real-time 3D Hand-Mesh Reconstruction

Xiao Tang Tianyu Wang Chi-Wing Fu
The Chinese University of Hong Kong
{xtang,wangty,cwfu}@cse.cuhk.edu.hk

Abstract

3D hand-mesh reconstruction from RGB images facilitates many applications, including augmented reality (AR). However, this requires not only real-time speed and accurate hand pose and shape but also plausible mesh-image alignment. While existing works already achieve promising results, meeting all three requirements is very challenging. This paper presents a novel pipeline by decoupling the hand-mesh reconstruction task into three stages: a joint stage to predict hand joints and segmentation; a mesh stage to predict a rough hand mesh; and a refine stage to fine-tune it with an offset mesh for mesh-image alignment. With careful design in the network structure and in the loss functions, we can promote high-quality finger-level mesh-image alignment and drive the models together to deliver real-time predictions. Extensive quantitative and qualitative results on benchmark datasets demonstrate that the quality of our results outperforms the state-of-the-art methods on hand-mesh/pose precision and hand-image alignment. In the end, we also showcase several real-time AR scenarios.

1. Introduction

3D hand-mesh reconstruction from a single monocular view is a long-standing task in computer vision that has great potentials for supporting and enhancing many applications, *e.g.*, human-computer interactions, augmented reality (AR), *etc.* By recognizing the 3D shape and pose of the user's hand in the AR view, we not only can augment the appearance of the hand and attach virtual objects onto the hand but can also enable the user to directly use his/her hand to grab and manipulate virtual 3D objects in the AR view. These are exciting but very challenging applications in AR that require the support of computer vision methods.

To put a hand-mesh reconstruction method into practice for direct hand interactions in AR, there are three requirements to meet. (i) The reconstruction should run in real time to give interactive feedback. (ii) The overall pose and shape of the reconstructed hand should match the user's real hand in AR. (iii) Beyond that, the reconstructed hand should plausibly align with the user's real hand in the image space to improve the perceptual realism in the AR interactions.

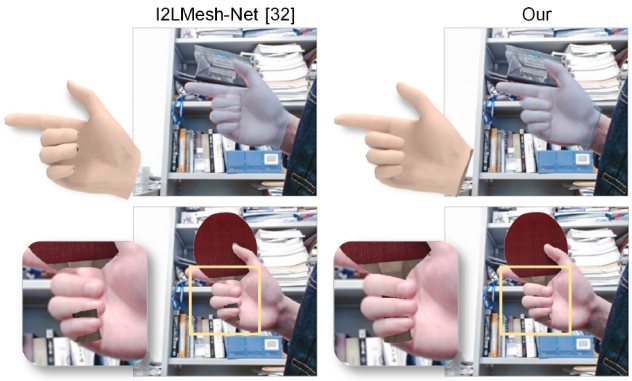

Figure 1. Comparing 3D hand meshes (top row) reconstructed by I2L-MeshNet [32] (ECCV 2020) and by our method. Taking these meshes to support AR interaction (bottom row), our predicted hand mesh delivers *better finger-level alignment* and *more natural hand occlusion* with the virtual ping pong racket.

At present, research works on hand-mesh reconstruction can roughly be divided into two categories based on the type of the input image, *i.e.*, RGB or depth. Despite the categorization, recent works are mostly deep-learning-based, *e.g.*, a typical approach is to use a deep neural network to predict the 2D/3D hand joint coordinates for guiding the regression of hand-mesh vertices [32] or parameters in a parametric hand model [60], *e.g.*, MANO [39]. Other works encode the image features to latent features and then use the Graph-CNN [15] or spiral filters [27] to directly reconstruct the hand mesh. Concerning the goal of supporting hand interactions in AR, existing works on hand-mesh reconstruction can mainly cater requirements (i) and (ii) but not requirement (iii), since they do not effectively utilize the 2D cues, *e.g.*, hand boundary, in their framework, while aiming for speed and gesture matching. A misaligned hand mesh may lead to obvious artifacts in the AR interactions; see the left column in Figure 1. While some recent works try to explore a better image-space alignment via differentiable rendering [1, 58] for meeting requirement (iii), they tend to struggle with requirement (i) due to the time-consuming iterative refinement process for the hand alignment.

978-1-6654-2812-5/21/$31.00 ©2021 IEEE
DOI 10.1109/ICCV48922.2021.01149

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.

The goal of this work is to simultaneously meet requirements (i)-(iii) for supporting real-time AR applications. Our key idea is to decouple the hand-mesh reconstruction process into three stages, especially to avoid the iterative refinement for hand alignment: (i) The *joint stage* encodes the input image and uses a multi-task architecture to predict hand segmentation and joints. During the testing, we only need to compute and pass image and joint features to the next stage. (ii) The *mesh stage* encodes the incoming features to quickly predict a rough 3D hand mesh with its lightweight architecture for speed. (iii) The *refine stage* extracts local features via the local feature projection unit and global features via the global feature broadcast unit for each vertex of the mesh and then utilizes a small Graph-CNN to predict an offset mesh to quickly align the rough mesh with the user's hand in the AR image space. In this decoupled design, offsets are essentially residuals; they are small vectors that the network can readily learn to regress based on the features gathered in the earlier stages. Besides, we utilize differentiable rendering to promote finger-level alignment.

We conduct extensive experiments to demonstrate that our proposed method boosts the hand-mesh reconstruction accuracy, comparing with the state-of-the-art. Also, the refine stage can efficiently produce a good image-mesh alignment for supporting AR applications. In the end, we further showcase several AR scenarios to demonstrate the potential of our method to support real-time AR interactions.

2. Related Work

3D hand-pose estimation aims to regress the 3D location of hand joints from a monocular RGB/depth image. Recent works are mainly deep-learning-based [61, 5, 31, 37, 56, 57, 28, 6, 9, 49, 2, 14, 22, 59, 44, 34, 17], and some [18, 47, 12] further predicted the hand-object pose. Among them, Li *et al.* [28] designed a binary selector that divides joints into groups and learns features independently for each group to avoid negative transfer. Cai *et al.* [4] trained a network with both synthetic data and weakly-labeled real RGB images. Very recently, Transformer [48] was adopted to further boost the hand-pose estimation precision [20, 21].

Depth-based 3D hand-mesh reconstruction aims to create the hand mesh from a depth image. Earlier works [45, 46, 24] often fitted a deformable hand mesh to the depth image with an iterative optimization. Recently, deep learning further helped to improve the performance. Malik *et al.* [30] adopted a CNN to regress the parameters of a linear-blend skinning model. Wan *et al.* [50] mapped features from a depth image to a mesh grid then sampled the mesh grid to recover the hand mesh. Malik *et al.* [29] voxelized the depth image and applied 3D convolutions to reconstruct the hand mesh. On the other hand, Mueller *et al.* [35] tracked bimanual hand interactions with a depth camera in real time.

RGB-based 3D hand-mesh reconstruction aims to reconstruct hand meshes using a commodity RGB camera [54, 33, 13, 38, 16, 51, 11, 42, 55, 8]. One common approach was to train a deep neural network with a parametric statistical model like MANO [39], which parameterizes a given triangular hand mesh with pose and shape parameters. Boukhayma *et al.* [3] regressed the MANO and view parameters to project the predicted MANO model to image space for supervision. Zhang *et al.* [58] and Baek *et al.* [1] adopted a differentiable rendering approach to supervise the training for silhouette alignment. Zhou *et al.* [60] first predicted the 3D coordinates of hand joints, then took the joint predictions as prior to produce the hand mesh via an inverse kinematics network. Moon *et al.* [32] introduced an image-to-lixel network to enhance the reconstruction accuracy.

To explicitly encode mesh structure in a deep neural network, Graph-CNN was widely adopted for hand-mesh reconstruction. Yet, Graph-CNN is designed for aggregating adjacent features based on the mesh topology, so it is less efficient for long-range features. To overcome this drawback, Ge *et al.* [15] and Choi *et al.* [10] proposed to regress the hand-mesh vertices using Graph-CNN in a coarse-to-fine manner. More recently, Kulon *et al.* [27] applied spiral filters for neighbourhood selection. Spurr *et al.* [43] proposed biomechanical constraints with weak supervision to effectively leverage additional 2D annotated images.

Existing works on hand-mesh reconstruction mostly focus on improving the precision of gesture prediction without efficiently utilizing 2D cues from the image. So these works are subpar for the hand-image alignment, which is, in fact, crucial for AR applications. In this work, we propose to decouple the hand-mesh estimation reconstruction, such that each stage can focus on a specific task for effective network learning and lightweight architecture. In this way, our approach can efficiently produce good-quality 3D hand meshes that align with the user's hand in the AR view, while delivering real-time performance.

3. Method

Figure 2 shows our three-stage framework for reconstructing the hand mesh from an input RGB photo:

(i) **The joint stage** encodes the input image and predicts a hand segmentation mask and hand joints J.

(ii) **The mesh stage** encodes features from the previous stage, including joint feature F_j from the joint decoder and shallow image feature F_l from the image encoder, and then predicts rough hand mesh M_r.

(iii) **The refine stage** aggregates the projected local and global features from the previous stages with the vertices in the rough mesh, then adopts a Graph-CNN (GCN) to regress offset mesh ΔM for producing the final hand mesh $M_f = M_r + \Delta M$.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.

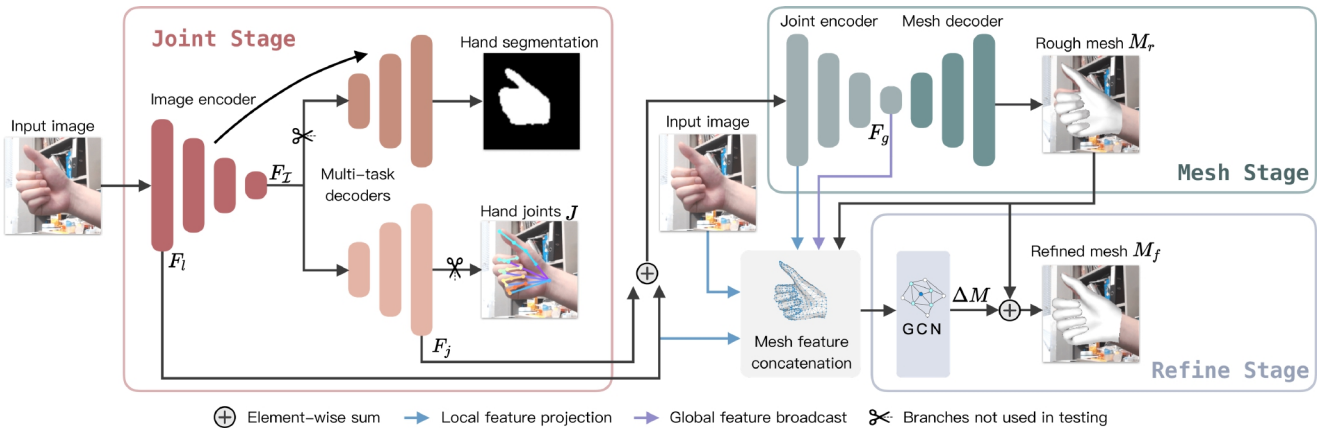

Figure 2. Our proposed framework. From the input image with the user's hand, the joint stage first extracts feature map $F_{\mathcal{I}}$ and sends it to two decoders, one for hand segmentation and the other for generating joint feature F_j and predicting the hand-joint locations J. Subsequently, the mesh stage fuses F_j and the shallow image feature F_l from the joint stage for predicting rough hand mesh M_r. Lastly, we aggregate image feature F_l from the joint stage with the shallow layers of the joint encoder and global feature F_g from the mesh stage, and passes the combined feature and the rough mesh to the graph convolutional layers (GCN) in the refine stage to regress offset mesh ΔM and produce the final hand mesh $M_f = M_r + \Delta M$. The scissor icon means the corresponding branch can be cut off during the testing.

Each stage in our framework has a clear goal, so they can *better focus on learning the associated features*, e.g., joint stage for the overall hand shape and joints, mesh stage for the rough hand mesh, and refine stage for learning to regress the *offset vectors to align the rough mesh with the user's hand in the image space*. Also, the network model in each stage can be kept *small and compact for achieving real-time performance*. Next, we present the three stages in detail.

3.1. Joint Stage

Given the input image, the joint stage first encodes it using a feature extractor and then feeds the encoded feature $F_{\mathcal{I}}$ into two branches: *one to predict the hand joints* and *the other to predict the hand segmentation*. For the hand-joint branch, we use a feature decoder to generate the joint feature map F_j and then regress the hand-joint locations, *i.e.*, 3D coordinates of the 21 joints of a hand (denoted as J), from F_j and $F_{\mathcal{I}}$ via multiple dilated convolutional layers and soft-argmax [7], following a similar strategy as [32]:

$$\begin{cases} J_x &= \text{soft-arg}\max(\text{Conv}_{\text{1D}}(avg^x(F_j))), \\ J_y &= \text{soft-arg}\max(\text{Conv}_{\text{1D}}(avg^y(F_j))), \\ J_z &= \text{Conv}_{\text{1D}}(\Phi(avg^{x,y}(F_{\mathcal{I}}))), \end{cases} \quad (1)$$

where subscripts x and y in J denote the image space and z denotes depth; Conv_{1D} denotes 1D convolution; Φ denotes a block, which consists of a fully connection layer, 1D batch normalization, and reshape function that rearranges the feature vector from $\mathbb{R}^C$ to $\mathbb{R}^{c' \times d}$; C is the channel size of feature vector; d is the size of $avg^x(F_l)$; and $c' = C/d$.

Considering that we need a plausible alignment between the reconstructed hand mesh and the user's hand in the image space, we thus harass shallow features from the image encoder to better capture the fine details of the hand, especially at the boundaries. Hence, we introduce another prediction head in the joint stage for the hand segmentation (see Figure 2), in which we use a U-Net [40] to process the encoded feature map and predict the hand segmentation.

3.2. Mesh Stage

In the mesh stage, we first fuse joint feature F_j and shallow image feature F_l from the joint stage through a joint encoder; see again Figure 2. Note that we choose F_j instead of J as input, since the predicted joints J may not be highly accurate, so it may misguide the reconstruction. Also, combining shallow feature F_l helps preserve the image features. The fused feature is then sent to an encoder unit to produce global mesh feature F_g. After that, we regress the 3D coordinates of the vertices in hand mesh M_r from F_g through multiple dilated convolutional layers and soft-argmax.

Since the rough mesh is restored from a low-resolution (1/64×1/64 of the original) global feature, in which most local features have lost due to the dilated kernels, so the result is mostly smooth and undersampled at boundaries. So, the rough mesh may not well align with the user's hand in the input image. Yet, this stage can quickly generate a rough hand mesh in 3D that captures the overall hand shape, while the refine stage can fine-tune the mesh by regressing per-vertex offset vectors that are small and easy to learn.

3.3. Refine Stage

The bottom right of Figure 2 illustrates how we residually refine the rough mesh M_r predicted by the previous mesh stage. Inspired by [25, 26, 52], we design the *local feature projection unit* and the *global feature broadcast unit*

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.

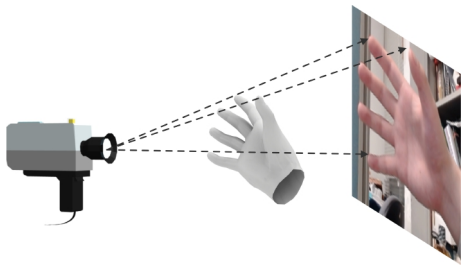

Figure 3. Through a 3D-to-2D projection from the 3D hand mesh to the 2D image space, we can extract high-resolution image features in the image feature space with a bilinear interpolation and collect image features for each vertex in the rough hand mesh.

to further extract local and global features from the input image and previous encoder layers, respectively.

Local feature projection unit. To produce a fine hand mesh that aligns with the real hand in input image, we are inspired by [25] to design the local feature projection unit to extract a single local feature vector per vertex in the rough mesh based on the previous shallow layers and input image.

Similiar to [52], we first project each mesh vertex to the 2D image space of the input image (equivalently, the spatial domain of feature maps) via a 3D-to-2D projection, as illustrated in Figure 3. Then, we perform a bilinear interpolation around each projected vertex on the feature maps to extract associated feature vectors. Here, the local feature projection unit collects features from the input image, the first layer of the image encoder, and also the first layer of the joint encoder; see the light blue arrows in Figure 2.

Global feature broadcast unit. The local features help us to address the fine details but they are insufficient. It is because they do not provide (global) information about the overall mesh structure. Concerning this, we introduce the global feature broadcast unit that broadcasts the mesh's global feature to every mesh vertex. Here, inspired by [26], we take deep feature F_g from the joint encoder in the mesh stage, apply a global average pooling over it to obtain a single 1-D vector, and reduce its channel dimension by 1/4 using a fully connected layer. After that, we attach this global feature vector to every vertex of the rough mesh; see the light purple arrow from F_g in Figure 2.

Graph-CNN. After collecting the local features from the local feature projection unit and global features from the global feature broadcast unit, we concatenate all the features at each vertex in the mesh and then send the concatenated mesh feature to our Graph-CNN. From a high-level perspective, the Graph-CNN aims to estimate one 3D offset vector for each mesh vertex based on its input 3D coordinates, as well as the collected local and global features, for aligning the rough mesh to the hand in the image. This processing is done by propagating features over the hand mesh topology. For the graph convolution layers, we adopt the

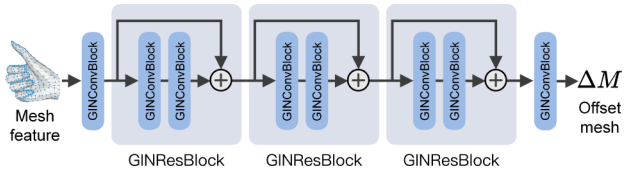

Figure 4. An illustration of our Graph-CNN network, whose input is the concatenated mesh feature (see Figure 2). This network starts with a GINConvBlock, followed by three GINResBlocks and another GINConvBlock, to produce the offset mesh ΔM.

formulation of Graph Isomorphism Network (GIN) convolution from [53], which is defined as:

$$x_i' = \text{MLPs}(x_i + \sum_{j \in \mathcal{N}(i)} x_j), \tag{2}$$

where x_i is the feature of the i-th node in the graph (equivalently the i-th vertex in the rough mesh); x_i' is the updated feature of the node; $\mathcal{N}(i)$ is an index set of the neighboring nodes of the i-th node; and MLPs denotes a series of multi-layer perceptrons. The framework of the Graph-CNN is illustrated in Figure 4. The network starts with a GINConvBlock, which contains a GIN convolution layer with ReLU and 1D batch normalization. After that, the network employs three GINResBlocks [26], each with two GINConvBlocks and an identity connection, followed by a GINConvBlock, to generate the offset mesh ΔM.

3.4. Training & Inference

We denote a dataset as $\{I^i, S^i, J^i_{gt}, M^i_{gt}\}_{i=1}^N$, where N is the total number of samples in the dataset; I^i is the i-th input image; S^i is the associated binary mask of the hand; and J^i_{gt} and M^i_{gt} are the associated annotated 3D joint and mesh coordinates of the hand, respectively. Note that the 3D joint coordinates can also be calculated from the mesh using a pre-defined regression matrix G, *e.g.*, $J^i_{gt} = GM^i_{gt}$.

We adopt an L1 loss to formulate both the mesh loss $\mathcal{L}_{\text{mesh}}$ and joint loss $\mathcal{L}_{\text{joint}}$ to supervise the predictions of mesh and joint in our three-stage framework:

$$\mathcal{L}_{\text{mesh}} = || M^i_{gt} - M^i_r || + || M^i_{gt} - M^i_f ||, \tag{3}$$

$$\begin{aligned}\mathcal{L}_{\text{joint}} = &|| J^i_{gt} - J^i || + || J^i_{gt} - GM^i_r || \\ &+ || J^i_{gt} - GM^i_f ||,\end{aligned} \tag{4}$$

Besides, inspired by [52], we adopt the normal loss $\mathcal{L}_{\text{norm}}$ for preserving the surface normals and the edge-length loss $\mathcal{L}_{\text{edge}}$ to penalize flying vertices:

$$\begin{aligned}\mathcal{L}_{\text{norm}} = &\sum_{f \in M^i_{gt}} \sum_{e \in f} || \langle \vec{e_r}, n^f_{gt} \rangle || \\ &+ \sum_{f \in M^i_{gt}} \sum_{e \in f} || \langle \vec{e_f}, n^f_{gt} \rangle ||\end{aligned} \tag{5}$$

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.

$$\text{and } \mathcal{L}_{\text{edge}} = \sum_{f \in M^i_{gt}} \sum_{e \in f} || \, |\vec{e_f}| - |\vec{e_{gt}}| \, || + \sum_{f \in M^i_{gt}} \sum_{e \in f} || \, |\vec{e_r}| - |\vec{e_{gt}}| \, ||, \tag{6}$$

where f denotes a triangle face; e denotes an edge of the triangle; $\vec{e_{gt}}$, $\vec{e_r}$, and $\vec{e_f}$ denote the edge vector on f that comes from M^i_{gt}, M^i_r, and M^i_f, respectively; and n^f_{gt} denotes the surface normal of f based on M^i_{gt}.

Next, we adopt the standard cross-entropy loss to supervise the hand segmentation:

$$\mathcal{L}_{\text{sil}} = -\sum_{j}^{H \times W} y^j \log p^j, \tag{7}$$

where $H \times W$ denotes the size of S^i; y^j denotes pixel j of S^i; and p^j denotes the prediction result of pixel j.

Lastly, we employ a differentiable renderer to render the predicted final hand mesh M_f in the image space to supervise the alignment through the render loss $\mathcal{L}_{\text{render}}$. Different from [58, 1] that use a binary silhouette mask in the supervision, we paint fingers of the ground-truth hand mesh with different colors and supervise the prediction by color matching to promote finger-level recognition. As the colors in the painted image indicate the occlusion relationship between the fingers and palm, "render" can further improve the joint prediction accuracy as well. Formally, we have

$$\mathcal{L}_{\text{render}} = \frac{1}{H_{\mathcal{R}} \times W_{\mathcal{R}}} \sum_{j}^{H_{\mathcal{R}} \times W_{\mathcal{R}}} || \, \mathcal{R}(M^i_f)^j - \mathcal{R}(M^i_{gt})^j \, ||^2, \tag{8}$$

where $\mathcal{R}$ denotes the differentiable renderer; $H_{\mathcal{R}} \times W_{\mathcal{R}}$ denotes the output resolution of $\mathcal{R}$; and $\mathcal{R}(M^i_f)^j$ and $\mathcal{R}(M^i_{gt})^j$ denotes the color of pixel j in $\mathcal{R}(M^i_f)$ and in $\mathcal{R}(M^i_{gt})$, respectively.

Our overall loss is $\mathcal{L} = \mathcal{L}_{\text{mesh}} + \lambda_j \mathcal{L}_{\text{joint}} + \lambda_n \mathcal{L}_{\text{normal}} + \lambda_e \mathcal{L}_{\text{edge}} + \lambda_s \mathcal{L}_{\text{sil}} + \lambda_r \mathcal{L}_{\text{render}}$, where we empirically set $\lambda_j = \lambda_n = \lambda_e = 1$, $\lambda_s = 10$, and $\lambda_r = 0.1$.

During the inference time, we can cut off the hand segmentation branch and the hand joints prediction in the joint stage to save computing time, without lowering the performance; see the scissor icons in Figure 2.

4. Experiment & Results

4.1. Experimental Settings

Datasets. (i) FreiHAND [62] contains 32,560 real training samples and 3,960 real test samples, all annotated with the MANO model. We use FreiHAND for both training and testing. Note that we randomly selected 2,000 samples from the training set for alignment evaluation and employed the rest for training. (ii) ObMan [18] has 141,550 synthetic training samples rendered with the MANO model. We also train our network on ObMan to further boost its performance. (iii) EgoDexter [36] has 1,485 real samples captured in an egocentric view; we use it only for testing.

Table 1. Quantitative comparisons between our method and state-of-the-arts on the FreiHAND dataset. ↓ means the lower the better; ↑ means the higher the better; and * means the method requires 2D joint annotations as input. The unit of ME and PE is 10mm.

Methods	venue	ME ↓	PE ↓	F@5mm ↑	F@15mm ↑	FPS
Mean shape [62]	ICCV2019	1.64	1.71	0.376	0.873	-
MANO fit [62]	ICCV2019	1.37	1.37	0.439	0.892	-
Hasson *et al.* [18]	ICCV2019	1.33	1.33	0.429	0.907	20
Boukh. *et al.* [3]	CVPR2019	1.32	3.50	0.427	0.894	10
ExPose [11]	ECCV2020	1.22	1.18	0.484	0.918	-
MANO CNN [62]	ICCV2019	1.09	1.10	0.516	0.934	-
Kulon *et al.* [27]	CVPR2020	0.86	0.84	0.614	0.966	60
I2L-MeshNet [32]	ECCV2020	0.76	0.74	0.681	0.973	33.3
Pose2Mesh* [10]	ECCV2020	0.76	0.74	0.683	0.973	22
Our w/o ObMan [18]		0.71	0.71	0.706	0.977	39.1
Our w/ ObMan [18]		**0.67**	**0.67**	**0.724**	**0.981**	39.1

Evaluation metrics. For quantitative evaluation, we follow the evaluation metrics in the FreiHAND online competition. (i) Mesh/pose error (ME/PE) measures the Euclidean distances between the predicted and ground-truth mesh vertices and joint coordinates. (ii) Mesh/Pose AUC reports the area under the curve of the percentage of correct key points (PCK) curve in different error threshold ranges. (iii) F-scores measures the harmonic mean of the recall and precision between the predicted and ground-truth vertices; here, we follow existing works to use F@5mm and F@15mm.

To evaluate image-space alignment, we compare the projected hand-mesh silhouette with the ground truth on two common segmentation metrics: (i) Mean Intersection over Union (mIoU), which measures the overlap regions and (ii) Hausdoff distance (HD), which is the maximum distance of a set to the nearest point in another set, so it is more sensitive at boundaries. We report 95% HD in the evaluation.

Implementation details. We used ResNet-50 [19] pre-trained on ImageNet [41] as the joint/mesh encoders. Our network was trained using the Adam optimizer on two Nvidia RTX Titan GPUs with a batch size of 32 per GPU for 25 epochs and an initial learning rate of $1e^{-4}$ (decay rate of 0.1 per 10 epochs). We resized the input image to 256×256 and augmented the data with random scaling, rotation, and color jittering. Our network runs at 39.1 frames per sec. on Nvidia RTX 2080 Ti, after we cut off the hand segmentation and joint prediction branches in joint stage. *Our code is released at* `https://wbstx.github.io/handar`.

4.2. Comparison with State-of-the-art Methods

We follow existing works [62, 18, 3, 11, 27, 32, 10] to quantitatively compare our method with state-of-the-art methods on the FreiHAND test set. Since the ground truths

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.

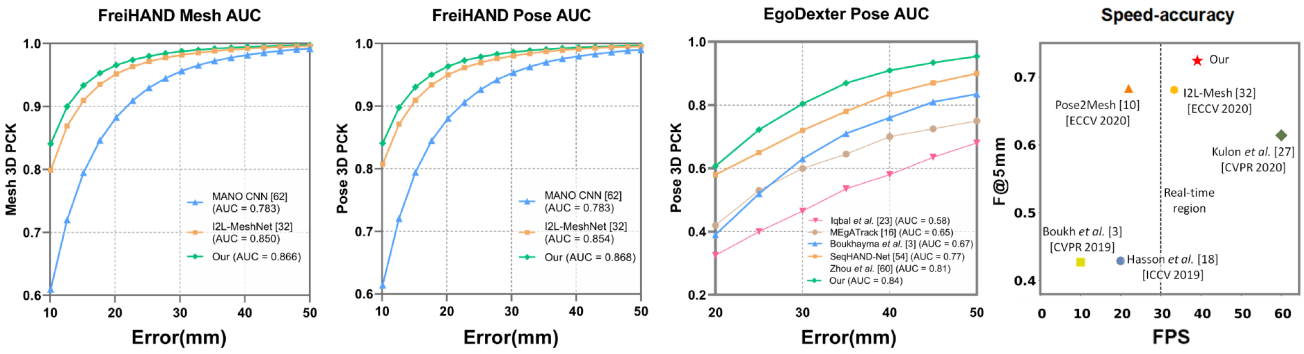

Figure 5. The mesh/pose AUC comparison with the state-of-the-art methods. From left to right, the first and second plots present the mesh and pose AUC results on the FreiHAND dataset, respectively, whereas the third plot presents the pose AUC result on the EgoDexter dataset. The fourth plot presents the speed-accuracy plot of 3D hand-mesh reconstruction on the FreiHAND test-set, in which we obtain the inference time by testing all methods on Nvidia RTX2080Ti-GPU. Our method achieves the best performance, while being real-time.

Figure 6. Visual comparisons between our method and state-of-the-arts. The input images (left column) are from the FreiHAND dataset [62]. Comparing the other three columns, we can see that our method predicts better hand meshes that match the gestures in input images with high-quality finger-level alignment.

of the test set are not accessible, the evaluation is conducted by submitting our test results to the online server. Table 1 and the left two plots of Figure 5 report the results, showing that our method outperforms others for all the aforementioned metrics. Further, Figure 6 shows visual comparisons on three test images in the dataset. For each input image, the first row shows the hand-image alignment by each method and the second row shows the predicted hand meshes. Please note that the code of Kulon *et al.* [27] has not been released and Pose2Mesh [10] requires 2D joints annotations as input, we omit them in the comparison. Comparing the results, we can see that other methods may fail to predict correct gestures or not able to well align the hand meshes in the image space. Our method is able to predict more accurate hand poses and shapes, and well aligns the hand meshes with the hands in images. More comparisons can be found in the supplemental material.

Next, we evaluate on an unseen dataset (not used in training), which is EgoDexter [36], to compare the generality of our method with several existing pose estimation methods [23, 3, 60, 16, 54]. Following [60], we use the centroids of the finger tips as roots to align the prediction and the ground truth. The third plot in Figure 5 shows the AUC result. Note that, since Zhou *et al.* [60] did not provide their pose PCK curve on EgoDexter, we only report their AUC. On the other hand, [32] did not test their performance on the EgoDexter dataset, so we do not report it in the figure. From the plot, we can see that our method achieves the highest pose AUC value, comparing with all the other methods, demonstrating its generality and potential for practical usage. Please check Figure 7 for more qualitative results from EgoDexter and FreiHAND. Also, we report the speed-accuracy comparison in the fourth plot shown in Figure 5. As we can see from this plot and also from Table 1, our method beats all recent methods on mesh prediction quality and also achieves real-time performance.

4.3. Comparison of Alignment

To further evaluate the alignment quality, we render the predicted hand mesh to produce a silhouette mask and compare it with the ground-truth silhouette using the aforementioned metrics. Here, we compare our method with I2L-MeshNet [32], which achieves state-of-the-art performance on the FreiHAND dataset without requiring 2D joint annotations. Since there are no ground-truth masks for the FreiHAND test set, we randomly select 2,000 images from

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.

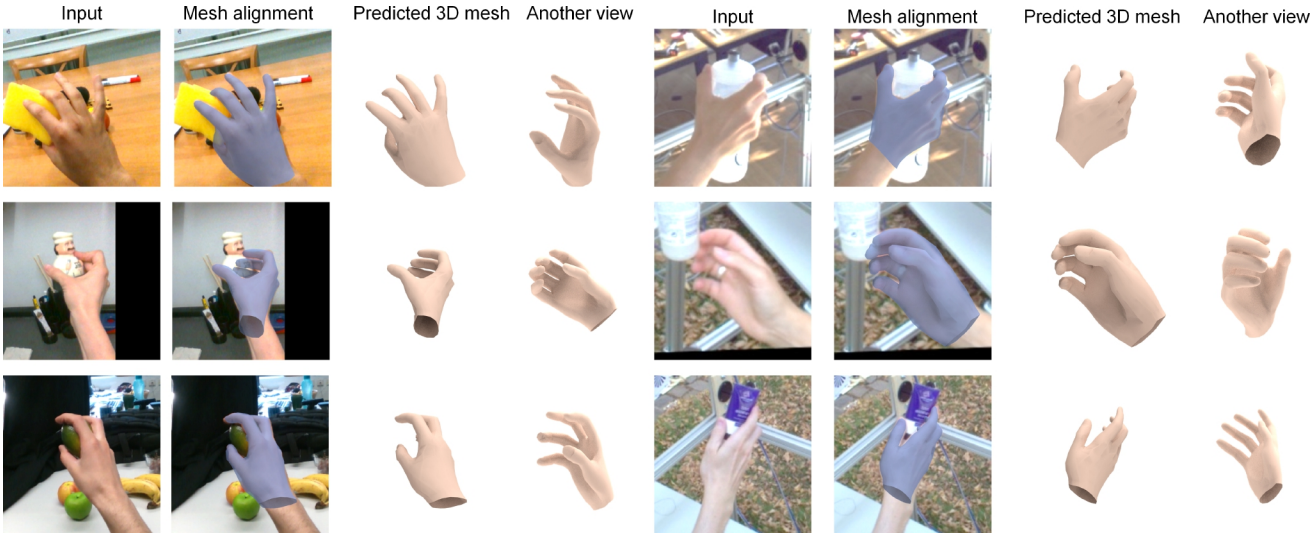

Figure 7. More qualitative results on test images in the EgoDexter dataset (left) and in the FreiHAND dataset (right).

Figure 8. The alignment evaluation is conducted on two levels. For hand-level alignment, we evaluate only the silhouette of the whole hand between the prediction and ground truth. For finger-level alignment, we evaluate individual fingers and palm. Finger-level alignment is more demanding than hand-level alignment.

Table 2. Quantitative alignment comparisons between our method and I2L-MeshNet [32]. ↓ means the lower the better and ↑ means the higher the better. Note that we report the average HD for the fingers and palm in the evaluation of finger-level alignment.

Methods	Hand-level		Finger-level	
	mIoU ↑	HD ↓	mIoU ↑	HD ↓
I2L-MeshNet [32]	92.08	6.17	71.07	8.42
Our w/o ObMan [18]	92.86	4.91	77.08	7.06
Our w/ ObMan [18]	**92.95**	**4.70**	**77.33**	**6.82**

the training set for validation and evaluate the hand-image alignment with resolution 224×224 on two levels: (i) hand-level alignment, in which we only evaluate the binary silhouette of the hand; and (ii) finger-level alignment, in which we evaluate the alignment for each finger and the palm. Note that finger-level alignment is more demanding than hand-level alignment; see Figure 8 for illustrations.

Table 2 shows the quantitative comparison result, from which we can see that our method outperforms I2LMeshNet [32] with a much higher mIoU and smaller HD values, which reveals a much better overall image-mesh alignment and also the accuracy at the boundaries.

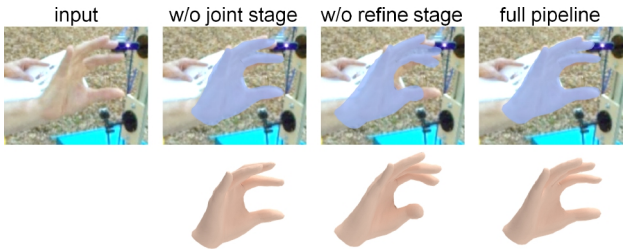

Figure 9. Ablation study on our decoupled design.

4.4. Ablation Study

Effects of decoupled design. To evaluate our decoupled design, we conducted an ablation study by simplifying our pipeline for the following two cases: (i) remove the joint stage and directly predict hand segmentation in the mesh stage; and (ii) remove the refine stage and treat M_r as the final hand mesh. The ablation results are reported in the top two rows of Table 3. We can observe obvious performance drop when removing any one of the two stages, showing the effectiveness of each stage and their contributions to the whole framework for better performance. Note that we conducted three runs and reported the average for each ablation case. Please see Figure 9 for visual comparison.

Effects of refinement unit & loss terms. Further, we evaluate the contributions of various units in the refine stage and in the losses for the following cases: (i) remove the local projection unit; (ii) remove the global broadcast unit; (iii) directly regress final mesh vertices by the Graph-CNN instead of offset vectors; (iv) remove the segmentation branch and $\mathcal{L}_{\text{sil}}$; and (v) remove $\mathcal{L}_{\text{render}}$. Table 3 reports the ablation results, in which we also conducted three runs and reported the average for each ablation case. Clearly, our full

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.

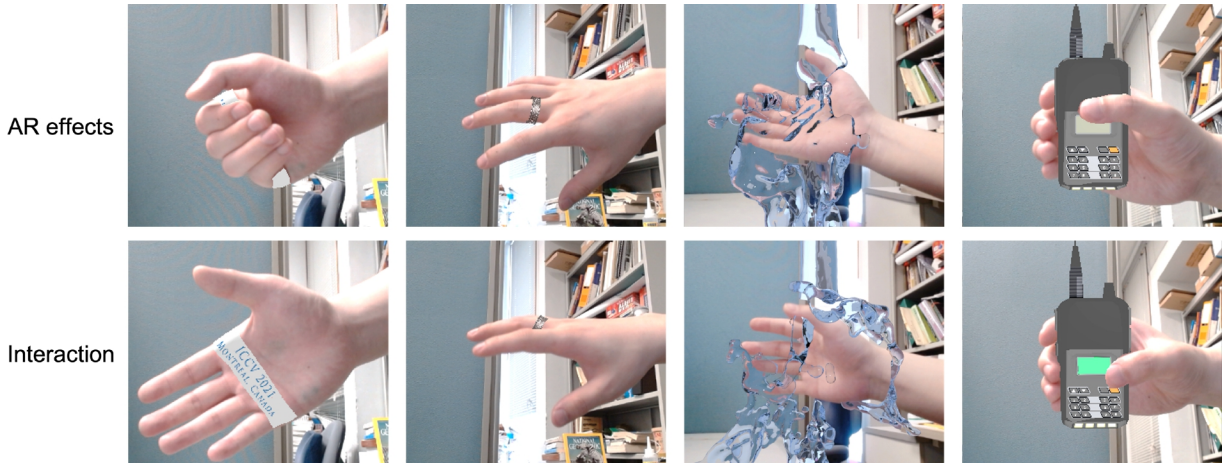

Figure 10. We showcase four example AR interaction scenarios to demonstrate our method's applicability (from left to right): (i) a virtual paper band around the user's hand, (ii) a virtual try-on with a ring (note the plausible hand occlusions with the virtual objects); (iii) we may splash virtual water on the user's hand and run a physical simulation, and (iv) the user can directly grab and manipulate a virtual walkie-talkie, and press a button on it. *Please watch full videos of these scenarios in* `https://wbstx.github.io/handar`.

Table 3. Comparing the performance of our full pipeline (bottom-most) with various ablation cases. For each case in the experiment, we conducted three runs and reported the average.

Models	Mesh AUC ↑	Pose AUC ↑	mIoU↑	HD↓
w/o joint stage	0.858±0	0.860±0	92.42±0.10	5.12±0.08
w/o refine stage	0.860±0	0.861±0	92.05±0.04	5.93±0.07
w/o local	0.860$\pm 5e^{-4}$	0.862$\pm 5e^{-4}$	92.55±0.05	5.34±0.04
w/o global	0.862$\pm 5e^{-4}$	0.864$\pm 5e^{-4}$	92.63±0.04	5.01±0.07
w/o offset	0.782$\pm 2e^{-3}$	0.858$\pm 3e^{-3}$	88.01±0.10	7.97±0.11
w/o $\mathcal{L}_{\text{sil}}$	0.862$\pm 5e^{-4}$	0.864$\pm 5e^{-4}$	92.50±0.08	5.19±0.08
w/o $\mathcal{L}_{\text{render}}$	0.863±0	0.865±0	92.80±0.02	4.97±0.04
Full	**0.866**$\pm 5e^{-4}$	**0.868**$\pm 5e^{-4}$	**92.95**±0.04	**4.70**±0.02

pipeline performs the best for all metrics, and removing any component reduces the overall performance, showing that each component contributes to improve the final result. Particularly, note that our method has a large performance drop when using the Graph-CNN to directly regress the mesh vertices instead of offset vectors. The reason behind is that our Graph-CNN is designed to regress small values, which are relatively easy to learn for this small Graph-CNN.

4.5. Applications

Our method can support direct hand interactions with 3D virtual objects in AR. Figures 1 and 10 show various interaction scenarios. With the reconstructed hand mesh that plausibly aligns with the user's hand in the input image, we can (i) resolve the occlusion between the virtual objects and the user's hand (see the first two examples in the figure); (ii) interact with a physical simulation, e.g., water, in the 3D AR space (see the third example); and (iii) directly grab and manipulate a virtual object, say by pressing on a button in the virtual walkie-talkie (see the last example in the figure). The associated full videos that were lively captured can be found in the supplemental material.

5. Conclusion & Future Work

This paper presents a new framework for 3D hand-mesh reconstruction by decoupling the task into three stages: the joint stage predicts the 3D coordinates of hand joints and the hand segmentation mask; the mesh stage estimates a rough 3D hand mesh; and the refine stage collects local and global features from previous layers and learns to regress per-vertex offset vectors to help align the rough mesh to the hand image with finger-level alignment. Experimental results demonstrate that our method outperforms the state-of-the-art methods for both the hand mesh/pose precision and mesh-image alignment. Also, our method is fast and can run in real-time on commodity graphics hardware.

In the future, we plan to explore methods to produce high-resolution (fine-grained) hand-mesh predictions that better match the smooth boundaries of the real hand. Also, we would like to explore the possibility of designing an adaptive refine stage that focuses more on aligning boundary vertices to improve the overall efficiency, while further cutting off the computation. Lastly, we plan also to explore the deployment of our method on mobile devices.

Acknowledgments. We thank anonymous reviewers for the valuable comments. This work is supported by the Research Grants Council of the Hong Kong Special Administrative Region (Project No. CUHK 14206320).

References

[1] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Pushing the envelope for RGB-based dense 3D hand pose estimation via neural rendering. In *CVPR*, pages 1067–1076, 2019.

[2] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Weakly-supervised domain adaptation via GAN and mesh

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.

model for estimating 3D hand poses interacting objects. In *CVPR*, pages 6121–6131, 2020.

[3] Adnane Boukhayma, Rodrigo de Bem, and Philip H.S. Torr. 3D hand shape and pose from images in the wild. In *CVPR*, pages 10843–10852, 2019.

[4] Yujun Cai, Liuhao Ge, Jianfei Cai, Nadia Magnenat-Thalmann, and Junsong Yuan. 3D hand pose estimation using synthetic data and weakly labeled RGB images. *IEEE PAMI*, 2020. to appear.

[5] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3D hand pose estimation from monocular RGB images. In *ECCV*, pages 666–682, 2018.

[6] Yujun Cai, Liuhao Ge, Jun Liu, Jianfei Cai, Tat-Jen Cham, Junsong Yuan, and Nadia Magnenat Thalmann. Exploiting spatial-temporal relationships for 3D pose estimation via graph convolutional networks. In *ICCV*, pages 2272–2281, 2019.

[7] Olivier Chapelle and Mingrui Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval*, pages 216–235, 2010.

[8] Xingyu Chen, Yufeng Liu, Chongyang Ma, Jianlong Chang, Huayan Wang, Tian Chen, Xiaoyan Guo, Pengfei Wan, and Wen Zheng. Camera-space hand mesh recovery via semantic aggregation and adaptive 2D-1D registration. In *CVPR*, pages 13274–13283, 2021.

[9] Yujin Chen, Zhigang Tu, Liuhao Ge, Dejun Zhang, Ruizhi Chen, and Junsong Yuan. SO-HandNet: Self-organizing network for 3D hand pose estimation with semi-supervised learning. In *ICCV*, pages 6961–6970, 2019.

[10] Hongsuk Choi, Gyeongsik Moon, and Kyoung Mu Lee. Pose2Mesh: Graph convolutional network for 3D human pose and mesh recovery from a 2D human pose. In *ECCV*, pages 769–787, 2020.

[11] Vasileios Choutas, Georgios Pavlakos, Timo Bolkart, Dimitrios Tzionas, and Michael J. Black. Monocular expressive body regression through body-driven attention. In *ECCV*, pages 20–40, 2020.

[12] Bardia Doosti, Shujon Naha, Majid Mirbagheri, and David J. Crandall. HOPE-Net: A graph-based model for hand-object pose estimation. In *CVPR*, pages 6608–6617, 2020.

[13] Zhipeng Fan, Jun Liu, and Yao Wang. Adaptive computationally efficient network for monocular 3D hand pose estimation. In *ECCV*, pages 127–144, 2020.

[14] Linpu Fang, Xingyan Liu, Li Liu, Hang Xu, and Wenxiong Kang. JGR-P2O: Joint graph reasoning based pixel-to-offset prediction network for 3D hand pose estimation from a single depth image. In *ECCV*, pages 120–137, 2020.

[15] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 3D hand shape and pose estimation from a single RGB image. In *CVPR*, pages 10833–10842, 2019.

[16] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, et al. MEgATrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics (SIGGRAPH)*, 39(4):87:1–87:13, 2020.

[17] Yana Hasson, Bugra Tekin, Federica Bogo, Ivan Laptev, Marc Pollefeys, and Cordelia Schmid. Leveraging photometric consistency over time for sparsely supervised hand-object reconstruction. In *CVPR*, pages 571–580, 2020.

[18] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *ICCV*, pages 11807–11816, 2019.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[20] Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I Yu. Epipolar transformers. In *CVPR*, pages 7779–7788, 2020.

[21] Lin Huang, Jianchao Tan, Ji Liu, and Junsong Yuan. Hand-Transformer: Non-autoregressive structured modeling for 3D hand pose estimation. In *ECCV*, pages 17–33, 2020.

[22] Weiting Huang, Pengfei Ren, Jingyu Wang, Qi Qi, and Haifeng Sun. AWR: Adaptive weighting regression for 3D hand pose estimation. In *AAAI*, pages 11061–11068, 2020.

[23] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5D heatmap regression. In *ECCV*, pages 118–134, 2018.

[24] Sameh Khamis, Jonathan Taylor, Jamie Shotton, Cem Keskin, Shahram Izadi, and Andrew Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *CVPR*, pages 2540–2548, 2015.

[25] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In *CVPR*, pages 9799–9808, 2020.

[26] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *CVPR*, pages 4501–4510, 2019.

[27] Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael Bronstein, and Stefanos Zafeiriou. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *CVPR*, pages 4990–5000, 2020.

[28] Shile Li and Dongheui Lee. Point-to-pose voting based hand pose estimation using residual permutation equivariant layer. In *CVPR*, pages 11927–11936, 2019.

[29] Jameel Malik, Ibrahim Abdelaziz, Ahmed Elhayek, Soshi Shimada, Sk Aziz Ali, Vladislav Golyanik, Christian Theobalt, and Didier Stricker. HandVoxNet: Deep voxel-based network for 3D hand shape and pose estimation from a single depth map. In *CVPR*, pages 7113–7122, 2020.

[30] Jameel Malik, Ahmed Elhayek, Fabrizio Nunnari, Kiran Varanasi, Kiarash Tamaddon, Alexis Heloir, and Didier Stricker. DeepHPS: End-to-end estimation of 3D hand pose and shape by learning from synthetic depth. In *3DV*, pages 110–119, 2018.

[31] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2V-PoseNet: Voxel-to-voxel prediction network for accurate 3D hand and human pose estimation from a single depth map. In *CVPR*, pages 5079–5088, 2018.

[32] 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*, pages 752–768, 2020.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.

[33] Gyeongsik Moon, Takaaki Shiratori, and Kyoung Mu Lee. DeepHandMesh: A weakly-supervised deep encoder-decoder framework for high-fidelity hand mesh modeling. In *ECCV*, pages 440–455, 2020.

[34] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. GANerated hands for real-time 3D hand tracking from monocular RGB. In *CVPR*, pages 49–59, 2018.

[35] Franziska Mueller, Micah Davis, Florian Bernard, Oleksandr Sotnychenko, Mickeal Verschoor, Miguel A Otaduy, Dan Casas, and Christian Theobalt. Real-time pose and shape reconstruction of two interacting hands with a single depth camera. *ACM Transactions on Graphics (SIGGRAPH)*, 38(4):49:1–49:13, 2019.

[36] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-time hand tracking under occlusion from an egocentric RGB-D sensor. In *ICCV*, pages 1154–1163, 2017.

[37] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. Using a single RGB frame for real time 3D hand pose estimation in the wild. In *WACV*, pages 436–445, 2018.

[38] Neng Qian, Jiayi Wang, Franziska Mueller, Florian Bernard, Vladislav Golyanik, and Christian Theobalt. HTML: A parametric hand texture model for 3D hand reconstruction and personalization. In *ECCV*, pages 54–71, 2020.

[39] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 36(6):245:1–245:17, 2017.

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015.

[41] 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*, 115(3):211–252, 2015.

[42] Jingjing Shen, Thomas J Cashman, Qi Ye, Tim Hutton, Toby Sharp, Federica Bogo, Andrew Fitzgibbon, and Jamie Shotton. The phong surface: Efficient 3D model fitting using lifted optimization. In *ECCV*, pages 687–703, 2020.

[43] Adrian Spurr, Umar Iqbal, Pavlo Molchanov, Otmar Hilliges, and Jan Kautz. Weakly supervised 3D hand pose estimation via biomechanical constraints. In *ECCV*, pages 211–228, 2020.

[44] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. Cross-modal deep variational hand pose estimation. In *CVPR*, pages 89–98, 2018.

[45] David Joseph Tan, Thomas Cashman, Jonathan Taylor, Andrew Fitzgibbon, Daniel Tarlow, Sameh Khamis, Shahram Izadi, and Jamie Shotton. Fits like a glove: Rapid and reliable hand shape personalization. In *CVPR*, pages 5610–5619, 2016.

[46] Jonathan Taylor, Richard Stebbing, Varun Ramakrishna, Cem Keskin, Jamie Shotton, Shahram Izadi, Aaron Hertzmann, and Andrew Fitzgibbon. User-specific hand modeling from monocular depth sequences. In *CVPR*, pages 644–651, 2014.

[47] Bugra Tekin, Federica Bogo, and Marc Pollefeys. H+O: Unified egocentric recognition of 3D hand-object poses and interactions. In *CVPR*, pages 4511–4520, 2019.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[49] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Self-supervised 3D hand pose estimation through training by fitting. In *CVPR*, pages 10853–10862, 2019.

[50] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Dual grid net: Hand mesh vertex regression from single depth maps. In *ECCV*, pages 442–459, 2020.

[51] Jiayi Wang, Franziska Mueller, Florian Bernard, Suzanne Sorli, Oleksandr Sotnychenko, Neng Qian, Miguel A. Otaduy, Dan Casas, and Christian Theobalt. RGB2Hands: real-time tracking of 3D hand interactions from monocular RGB video. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 39(6):218:1–218:16, 2020.

[52] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV*, pages 52–67, 2018.

[53] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[54] John Yang, Hyung Jin Chang, Seungeui Lee, and Nojun Kwak. SeqHAND: RGB-sequence-based 3D hand pose and shape estimation. In *ECCV*, pages 122–139, 2020.

[55] Lixin Yang, Jiasen Li, Wenqiang Xu, Yiqun Diao, and Cewu Lu. BiHand: Recovering hand mesh with multi-stage bisected hourglass networks. In *BMVC*, 2020.

[56] Linlin Yang, Shile Li, Dongheui Lee, and Angela Yao. Aligning latent spaces for 3D hand pose estimation. In *ICCV*, pages 2335–2343, 2019.

[57] Linlin Yang and Angela Yao. Disentangling latent hands for image synthesis and pose estimation. In *CVPR*, pages 9877–9886, 2019.

[58] Xiong Zhang, Qiang Li, Hong Mo, Wenbo Zhang, and Wen Zheng. End-to-end hand mesh recovery from a monocular RGB image. In *ICCV*, pages 2354–2364, 2019.

[59] Long Zhao, Xi Peng, Yuxiao Chen, Mubbasir Kapadia, and Dimitris N. Metaxas. Knowledge as priors: Cross-modal knowledge generalization for datasets without superior knowledge. In *CVPR*, pages 6528–6537, 2020.

[60] Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. Monocular real-time hand shape and motion capture using multi-modal data. In *CVPR*, pages 5346–5355, 2020.

[61] Christian Zimmermann and Thomas Brox. Learning to estimate 3D hand pose from single RGB images. In *ICCV*, pages 4903–4911, 2017.

[62] 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*, pages 813–822, 2019.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 03:03:32 UTC from IEEE Xplore. Restrictions apply.