3D Hand Shape and Pose Estimation from a Single RGB Image

Liuhao Ge[1]*, Zhou Ren[4], Yuncheng Li[2], Zehao Xue[2], Yingying Wang[2], Jianfei Cai[1], Junsong Yuan[3]
[1]Nanyang Technological University [2]Snap Inc.
[3]State University of New York at Buffalo [4]Wormpex AI Research
ge0001ao@e.ntu.edu.sg, zhou.ren@bianlifeng.com, yuncheng.li@snap.com,
zehao.xue@snap.com, ywang@snap.com, asjfcai@ntu.edu.sg, jsyuan@buffalo.edu

Abstract

This work addresses a novel and challenging problem of estimating the full 3D hand shape and pose from a single RGB image. Most current methods in 3D hand analysis from monocular RGB images only focus on estimating the 3D locations of hand keypoints, which cannot fully express the 3D shape of hand. In contrast, we propose a Graph Convolutional Neural Network (Graph CNN) based method to reconstruct a full 3D mesh of hand surface that contains richer information of both 3D hand shape and pose. To train networks with full supervision, we create a large-scale synthetic dataset containing both ground truth 3D meshes and 3D poses. When fine-tuning the networks on real-world datasets without 3D ground truth, we propose a weakly-supervised approach by leveraging the depth map as a weak supervision in training. Through extensive evaluations on our proposed new datasets and two public datasets, we show that our proposed method can produce accurate and reasonable 3D hand mesh, and can achieve superior 3D hand pose estimation accuracy when compared with state-of-the-art methods.

1. Introduction

Vision-based 3D hand analysis is a very important topic because it has many applications in virtual reality (VR) and augmented reality (AR). However, despite years of studies [40, 27, 57, 58, 47, 45, 12, 26], it remains an open problem due to the diversity and complexity of hand shape, pose, gesture, occlusion, *etc*. In the past decade, we have witnessed a rapid advance in 3D hand pose estimation from depth images [35, 52, 11, 14, 13, 61, 10, 15]. Considering RGB cameras are more widely available than depth cameras, some recent works start looking into 3D hand analysis from monocular RGB images, and mainly focus on estimating sparse 3D hand joint locations but ignore dense 3D hand shape [63, 44, 32, 4, 19, 36, 38]. However, many immersive VR and AR applications often require accurate estimation of both 3D hand pose and 3D hand shape.

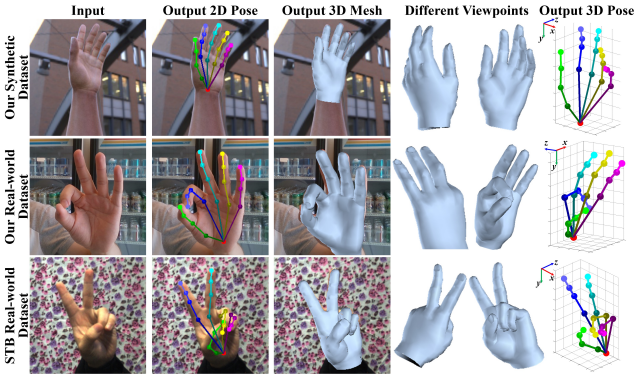

Figure 1: Our proposed method is able to not only estimate 2D/3D hand joint locations, but also recover a full 3D mesh of hand surface from a single RGB image. We show our estimation results on our proposed synthetic and real-world datasets as well as the STB real-world dataset [62].

This motivates us to bring out a more challenging task: *how to jointly estimate not only the 3D hand joint locations, but also the full 3D mesh of hand surface from a single RGB image?* In this work, we develop a sound solution to this task, as illustrated in Fig. 1.

The task of single-view 3D hand shape estimation has been studied previously, but mostly in controlled settings, where a depth sensor is available. The basic idea is to fit a generative 3D hand model to the input depth image with iterative optimization [49, 30, 23, 20, 51, 41]. In contrast, here we consider to estimate 3D hand shape from a monocular RGB image, which has not been extensively studied yet. The absence of explicit depth cues in RGB images makes this task difficult to be solved by iterative optimization approaches. In this work, we apply deep neural networks that are trained in an end-to-end manner to recover 3D hand mesh directly from a single RGB image. Specifically, we predefine the topology of a triangle mesh representing the

*This work is done when Liuhao Ge is a research intern at Snap Inc.

hand surface, and aim at estimating the 3D coordinates of all the vertices in the mesh using deep neural networks. To achieve this goal, there are several challenges.

The first challenge is the high dimensionality of the output space for 3D hand mesh generation. Compared with estimating sparse 3D joint locations of the hand skeleton (*e.g.*, 21 joints), it is much more difficult to estimate 3D coordinates of dense mesh vertices (*e.g.*, 1280 vertices) using conventional CNNs. One straightforward solution is to follow the common approach used in human body shape estimation [53, 48, 37, 21], namely to regress low-dimensional parameters of a predefined deformable hand model, *e.g.*, MANO [42]. However, due to the small amount of training data and the linear bases utilized for reconstruction [42], the representation capacity of MANO model could be limited for nonlinear variations in hand shape.

In this paper we argue that the output 3D hand mesh vertices in essence are graph-structured data, since a 3D mesh can be easily represented as a graph. To output such graph-structured data and better exploit the topological relationship among mesh vertices in the graph, motivated by recent works on Graph CNNs [7, 39, 56], we propose a novel Graph CNN-based approach. Specifically, we adopt graph convolutions [7] hierarchically with upsampling and nonlinear activations to generate 3D hand mesh vertices in a graph from image features which are extracted by backbone networks. With such an end-to-end trainable framework, our Graph CNN-based method can better represent the highly variable 3D hand shapes, and can better express the local details of 3D hand shapes.

Besides the computational model, an additional challenge is the lack of ground truth 3D hand mesh training data for real-world images. Manually annotating the ground truth 3D hand meshes on real-world RGB images is extremely laborious and time-consuming. We thus choose to create a large-scale synthetic dataset containing the ground truth of both 3D hand mesh and 3D hand pose for training. However, models trained on the synthetic dataset usually produce unsatisfactory estimation results on real-world datasets due to the domain gap between them. To address this issue, inspired by [4, 37], we propose a novel weakly-supervised method by leveraging depth map as a weak supervision for 3D mesh generation, since depth map can be easily captured by an RGB-D camera when collecting real-world training data. More specifically, when fine-tuning on real-world datasets, we render the generated 3D hand mesh to a depth map on the image plane and minimize the depth map loss against the reference depth map, as shown in Fig. 3. Note that, during testing, we only need an RGB image as input to estimate full 3D hand shape and pose.

To the best of our knowledge, we are the first to handle the problem of estimating not only 3D hand pose but also full 3D hand shape from a single RGB image. Our main contributions are summarized as follows:

- We propose a novel end-to-end trainable hand mesh generation approach based on Graph CNN [7]. Compared with MANO [42] based model, our method can better represent hand shape variations and capture local details. Furthermore, we observe that by estimating full 3D hand mesh, our method boost the accuracy performance of 3D hand pose estimation, as validated in Sec. 5.4.
- We propose a weakly-supervised training pipeline on real-world dataset, by rendering the generated 3D mesh to a depth map on the image plane and leveraging the reference depth map as a weak supervision, without requiring any annotations of 3D hand mesh or 3D hand pose for real-world images.
- We introduce the first large-scale synthetic RGB-based 3D hand shape and pose dataset as well as a small-scale real-world dataset, which contain the annotation of both 3D hand joint locations and the full 3D meshes of hand surface. We will share our datasets publicly upon the acceptance of this work.

We conduct comprehensive experiments on our proposed synthetic and real-world datasets as well as two public datasets [62, 63]. Experimental results show that our proposed method can produce accurate and reasonable 3D hand mesh with real-time speed on GPU, and can achieve superior accuracy performance on 3D hand pose estimation when compared with state-of-the-art methods.

2. Related Work

3D hand shape and pose estimation from depth images: Most previous methods estimate 3D hand shape and pose from depth images by fitting a deformable hand model to the input depth map with iterative optimization [49, 30, 23, 20, 51, 41]. A recent method [31] was proposed to estimate pose and shape parameters from the depth image using CNNs, and recover 3D hand meshes using LBS. The CNNs are trained in an end-to-end manner with mesh and pose losses. However, the quality of their recovered hand meshes is restricted by their simple LBS model.

3D hand pose estimation from RGB images: Pioneering works [58, 6] estimate hand pose from RGB image sequences. Gorce et al. [6] proposed estimating 3D hand pose, the hand texture and the illuminant dynamically through minimization of an objective function. Sridhar et al. [46] adopted multi-view RGB images and depth data to estimate the 3D hand pose by combining a discriminative method with local optimization. With the advance of deep learning and the wide applications of monocular RGB cameras, many recent works estimate 3D hand pose from a single RGB image using deep neural networks [63, 44, 32, 4, 19,

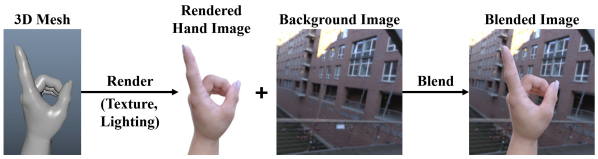

Figure 2: Illustration of our synthetic hand shape and pose dataset creation as well as background image augmentation during training.

38]. However, few works focus on 3D hand shape estimation from RGB images. Panteleris *et al*. [36] proposed to fit a 3D hand model to the estimated 2D joint locations. But the hand model is controlled by 27 hand pose parameters, thus it cannot well adapt to various hand shapes. In addition, this method is not an end-to-end framework for generating 3D hand mesh.

3D human body shape and pose estimation from a single RGB image: Most recent methods rely on SMPL, a body shape and pose model [29]. Some methods fit the SMPL model to the detected 2D keypoints [3, 24]. Some methods regress SMPL parameters using CNNs with supervisions of silhouette and/or 2D keypoints [48, 37, 21]. A more recent method [54] predicts a volumetric representation of human body. Different from these methods, we propose to estimate 3D mesh vertices using Graph CNNs in order to learn nonlinear hand shape variations and better utilize the relationship among vertices in the mesh topology. In addition, instead of using 2D silhouette or 2D keypoints to weakly supervise the network training, we propose to leverage the depth map as a weak 3D supervision when training on real-world datasets without 3D mesh or 3D pose annotations.

3. 3D Hand Shape and Pose Dataset Creation

Manually annotating the ground truth of 3D hand meshes and 3D hand joint locations for real-world RGB images is extremely laborious and time-consuming. To overcome the difficulties in real-world data annotation, some works [43, 63, 33] have adopted synthetically generated hand RGB images for training. However, existing hand RGB image datasets [43, 62, 63, 33] only provide the annotations of 2D/3D hand joint locations, and they do not contain any 3D hand shape annotations. Thus, these datasets are not suitable for the training of the 3D hand shape estimation task.

In this work, we create a large-scale synthetic hand shape and pose dataset that provides the annotations of both 3D hand joint locations and full 3D hand meshes. In particular, we use Maya [2] to create a 3D hand model and rig it with joints, and then apply photorealistic textures on it as well as natural lighting using High-Dynamic-Range (HDR) images. We model hand variations by creating blend shapes with different shapes and ratios, then applying random weights on the blend shapes. To fully explore the pose space, we create hand poses from 500 common hand gestures and 1000 unique camera viewpoints. To simulate real-world diversity, we use 30 lightings and five skin colors. We render the hand using global illumination with off-the-shelf Arnold renderer [1]. The rendering tasks are distributed onto a cloud render farm for maximum efficiency. In total, our synthetic dataset contains 375,000 hand RGB images with large variations. We use 315,000 images for training and 60,000 images for validation. During training, we randomly sample and crop background images from COCO [28], LSUN [60], and Flickr [9] datasets, and blend them with the rendered hand images, as shown in Fig. 2.

In addition, to quantitatively evaluate the performance of hand mesh estimation on real-world image, we create a real-world dataset containing 583 hand RGB images with the annotations of 3D hand mesh and 3D hand joint locations. To facilitate the 3D annotation, we capture the corresponding depth images using an Intel RealSense RGB-D camera [18] and manually adjust the 3D hand model in Maya with the reference of both RGB images and depth points. In this work, this real-world dataset is only used for evaluation.

4. Methodology

4.1. Overview

We propose to generate a full 3D mesh of the hand surface and the 3D hand joint locations directly from a single monocular RGB image, as illustrated in Fig. 3. Specifically, the input is a single RGB image centered on a hand, which is passed through a two-stacked hourglass network [34] to infer 2D heat-maps. The estimated 2D heat-maps, combined with the image feature maps, are encoded as a latent feature vector by using a residual network [17] that contains eight residual layers and four max pooling layers. The encoded latent feature vector is then input to a Graph CNN [7] to infer the 3D coordinates of N vertices $\mathcal{V} = \{\boldsymbol{v}_i\}_{i=1}^{N}$ in the 3D hand mesh. The 3D hand joint locations $\boldsymbol{\Phi} = \{\boldsymbol{\phi}_j\}_{j=1}^{J}$ are linearly regressed from the reconstructed 3D hand mesh vertices by using a simplified linear Graph CNN.

In this work, we first train the network models on a synthetic dataset and then fine-tune them on real-world datasets. On the synthetic dataset that contains the ground truth of 3D hand meshes and 3D hand joint locations, we train the networks end-to-end in a fully-supervised manner by using 2D heat-map loss, 3D mesh loss, and 3D pose loss. More details will be presented in Section 4.3. On the real-world dataset, the networks can be fine-tuned in a weakly-supervised manner without requiring the ground truth of 3D hand meshes or 3D hand joint locations. To achieve this target, we leverage the reference depth map available in training, which can be easily captured from a depth camera, as a weak supervision during the fine-tuning,

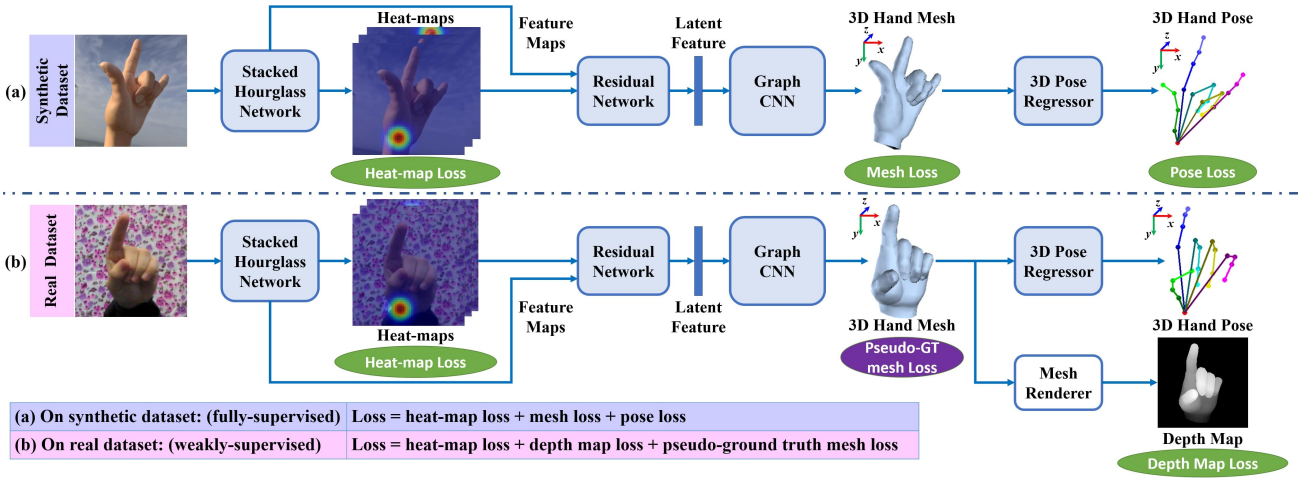

Figure 3: Overview of our method for 3D hand shape and pose estimation from a single RGB image. Our network model is first trained on a synthetic dataset in a fully supervised manner with heat-map loss, 3D mesh loss, and 3D pose loss, as shown in (a); and then fine-tuned on a real-world dataset without 3D mesh or 3D pose ground truth in a weakly-supervised manner by innovatively introducing a pseudo-ground truth mesh loss and a depth map loss, as shown in (b). For both (a) and (b), the input RGB image is first passed through a two-stacked hourglass network [34] for extracting feature maps and 2D heat-maps, which are then combined and encoded as a latent feature vector by a residual network [17]. The latent feature is fed into a Graph CNN [7] to infer the 3D coordinates of mesh vertices. Finally, the 3D hand pose is linearly regressed from the 3D hand mesh. During training on the real-world dataset, as shown in (b), the generated 3D hand mesh is rendered to a depth map to compute the depth map loss against the reference depth map. Note that this step is not involved in testing.

and employ a differentiable renderer to render the generated 3D mesh to a depth map from the camera viewpoint. To guarantee the mesh quality, we generate the pseudo-ground truth mesh from the pretrained model as an additional supervision. More details will be presented in Section 4.4.

4.2. Graph CNNs for Mesh and Pose Estimation

Graph CNNs have been successfully applied in modeling graph structured data [56, 59, 55]. As 3D hand mesh is of graph structure by nature, in this work we adopt the Chebyshev Spectral Graph CNN [7] to generate 3D coordinates of vertices in the hand mesh and estimate 3D hand pose from the generated mesh.

A 3D mesh can be represented by an undirected graph $\mathcal{M} = (\mathcal{V}, \mathcal{E}, W)$, where $\mathcal{V} = \{\boldsymbol{v}_i\}_{i=1}^{N}$ is a set of N vertices in the mesh, $\mathcal{E} = \{\boldsymbol{e}_i\}_{i=1}^{E}$ is a set of E edges in the mesh, $W = (w_{ij})_{N\times N}$ is the adjacency matrix, where $w_{ij} = 0$ if $(i,j) \notin \mathcal{E}$, and $w_{ij} = 1$ if $(i,j) \in \mathcal{E}$. The normalized graph Laplacian [5] is computed as $L = I_N - D^{-1/2} W D^{-1/2}$, where $D = \mathrm{diag}\left(\sum_j w_{ij}\right)$ is the diagonal degree matrix, I_N is the identity matrix. Here, we assume that the topology of the triangular mesh is fixed and is predefined by the hand mesh model, *i.e.*, the adjacency matrix W and the graph Laplacian L of the graph $\mathcal{M}$ are fixed during training and testing.

Given a signal $\mathbf{f} = (f_1, \cdots, f_N)^T \in \mathbb{R}^{N\times F}$ on the vertices of graph $\mathcal{M}$, it represents F-dim features of N vertices in the 3D mesh. In Chebyshev Spectral Graph CNN [7], the graph convolutional operation on a graph signal $\mathbf{f}_{\mathrm{in}} \in \mathbb{R}^{N\times F_{\mathrm{in}}}$ is defined as

$$\mathbf{f}_{\mathrm{out}} = \sum_{k=0}^{K-1} T_k\left(\tilde{L}\right) \cdot \mathbf{f}_{\mathrm{in}} \cdot \theta_k, \tag{1}$$

where $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ is the Chebyshev polynomial of degree k, $T_0 = 1$, $T_1 = x$; $\tilde{L} \in \mathbb{R}^{N\times N}$ is the rescaled Laplacian, $\tilde{L} = 2L/\lambda_{\max} - I_N$, $\lambda_{\max}$ is the maximum eigenvalue of L; $\theta_k \in \mathbb{R}^{F_{\mathrm{in}}\times F_{\mathrm{out}}}$ are the trainable parameters in the graph convolutional layer; $\mathbf{f}_{\mathrm{out}} \in \mathbb{R}^{N\times F_{\mathrm{out}}}$

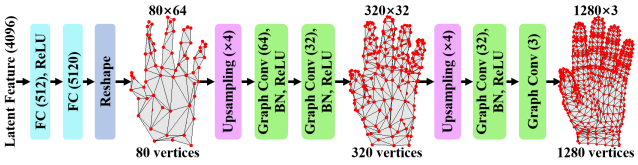

Figure 4: Architecture of the Graph CNN for mesh generation. The input is a latent feature vector extracted from the input RGB image. Passing through two fully-connected (FC) layers, the feature vector is transformed into 80 vertices with 64-dim features in a coarse graph. The features are upsampled and allocated to a finer graph. With two upsampling layers and four graph convolutional layers, the network outputs 3D coordinates of the 1280 mesh vertices. The numbers in parentheses of FC layers and graph convolutions represent the dimensions of output features.

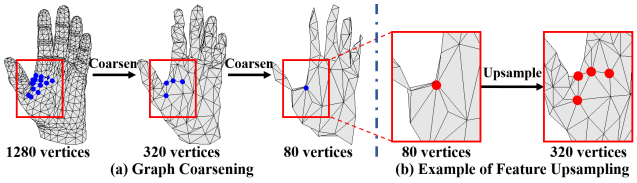

Figure 5: (a) Given our predefined mesh topology, we first perform graph coarsening [7] to cluster meaningful neighborhoods on graphs and create a tree structure to store correspondences of vertices in graphs at adjacent coarsening levels. (b) During the forward propagation, we perform feature upsampling. The feature of a vertex in the coarse graph is allocated to its children vertices in the finer graph.

is the output graph signal. This operation is K-localized since Eq. 1 is a K-order polynomial of the graph Laplacian, and it only affects the K-hop neighbors of each central node. Readers are referred to [7] for more details.

In this work, we design a hierarchical architecture for mesh generation by performing graph convolution on graphs from coarse to fine, as shown in Fig. 4. The topologies of coarse graphs are precomputed by graph coarsening, as shown in Fig. 5 (a), and are fixed during training and testing. Following Defferrard *et al.* [7], we use the Graclus multilevel clustering algorithm [8] to coarsen the graph, and create a tree structure to store correspondences of vertices in graphs at adjacent coarsening levels. During the forward propagation, we upsample features of vertices in the coarse graph to corresponding children vertices in the fine graph, as shown in Fig. 5 (b). Then, we perform the graph convolution to update features in the graph. All the graph convolutional filters have the same support of $K = 3$. To make the network output irrelevant to the camera intrinsic parameters, we design the network to output UV coordinates on input image and depth of vertices in the mesh, which can be converted to 3D coordinates in the camera coordinate system using the camera intrinsic matrix. Similar to [63, 4, 44], we estimate scale-invariant and root-relative depth of mesh vertices.

Considering that 3D joint locations can be estimated directly from the 3D mesh vertices using a linear regressor [29, 42], we adopt a simplified Graph CNN [7] with two pooling layers and without nonlinear activation to linearly regress the scale-invariant and root-relative 3D hand joint locations from 3D coordinates of hand mesh vertices.

4.3. Fully-supervised Training on Synthetic Dataset

We first train the networks on our synthetic hand shape and pose dataset in a fully-supervised manner. As shown in Fig. 3 (a), the networks are supervised by heat-map loss $\mathcal{L}_{\mathcal{H}}$, mesh loss $\mathcal{L}_{\mathcal{M}}$, and 3D pose loss $\mathcal{L}_{\mathcal{J}}$.

Heat-map Loss. $\mathcal{L}_{\mathcal{H}} = \sum_{j=1}^{J} \left\| \mathcal{H}_j - \hat{\mathcal{H}}_j \right\|_2^2$, where $\mathcal{H}_j$ and $\hat{\mathcal{H}}_j$ are the ground truth and estimated heat-maps, respectively. We set the heat-map resolution as 64×64 px. The ground truth heat-map is defined as a 2D Gaussian with a standard deviation of 4 px centered on the ground truth 2D joint location.

Mesh Loss. Similar to [56], $\mathcal{L}_{\mathcal{M}} = \lambda_v \mathcal{L}_v + \lambda_n \mathcal{L}_n + \lambda_e \mathcal{L}_e + \lambda_l \mathcal{L}_l$ is composed of vertex loss $\mathcal{L}_v$, normal loss $\mathcal{L}_n$, edge loss $\mathcal{L}_e$, and Laplacian loss $\mathcal{L}_l$. The vertex loss $\mathcal{L}_v$ is to constrain 2D and 3D locations of mesh vertices:

$$\mathcal{L}_v = \sum_{i=1}^{N} \left\| \boldsymbol{v}_i^{3D} - \hat{\boldsymbol{v}}_i^{3D} \right\|_2^2 + \left\| \boldsymbol{v}_i^{2D} - \hat{\boldsymbol{v}}_i^{2D} \right\|_2^2, \quad (2)$$

where $\boldsymbol{v}_i$ and $\hat{\boldsymbol{v}}_i$ denote the ground truth and estimated 2D/3D locations of the mesh vertices, respectively. The normal loss $\mathcal{L}_n$ is to enforce surface normal consistency:

$$\mathcal{L}_n = \sum_t \sum_{(i,j)\in t} \left\| \langle \hat{\boldsymbol{v}}_i^{3D} - \hat{\boldsymbol{v}}_j^{3D}, \boldsymbol{n}_t \rangle \right\|_2^2, \quad (3)$$

where t is the index of triangle faces in the mesh; (i, j) are the indices of vertices that compose one edge of triangle t; and $\boldsymbol{n}_t$ is the ground truth normal vector of triangle face t, which is computed from ground truth vertices. The edge loss $\mathcal{L}_e$ is introduced to enforce edge length consistency:

$$\mathcal{L}_e = \sum_{i=1}^{E} \left(\|\boldsymbol{e}_i\|_2^2 - \|\hat{\boldsymbol{e}}_i\|_2^2 \right)^2, \quad (4)$$

where $\boldsymbol{e}_i$ and $\hat{\boldsymbol{e}}_i$ denote the ground truth and estimated edge vectors, respectively. The Laplacian loss $\mathcal{L}_l$ is introduced to preserve the local surface smoothness of mesh:

$$\mathcal{L}_l = \sum_{i=1}^{N} \left\| \boldsymbol{\delta}_i - \sum_{\boldsymbol{v}_k \in \mathcal{N}(\boldsymbol{v}_i)} \boldsymbol{\delta}_k \Big/ B_i \right\|_2^2, \quad (5)$$

where $\boldsymbol{\delta}_i = \boldsymbol{v}_i^{3D} - \hat{\boldsymbol{v}}_i^{3D}$ is the offset from the estimation to the ground truth, $\mathcal{N}(\boldsymbol{v}_i)$ is the set of neighboring vertices of $\boldsymbol{v}_i$, and B_i is the number of vertices in the set $\mathcal{N}(\boldsymbol{v}_i)$. This loss function prevents the neighboring vertices from having opposite offsets, thus making the estimated 3D hand surface mesh smoother. For the hyperparameters, we set $\lambda_v = 1$, $\lambda_n = 1$, $\lambda_e = 1$, $\lambda_l = 50$ in our implementation.

3D Pose Loss. $\mathcal{L}_{\mathcal{J}} = \sum_{j=1}^{J} \left\| \boldsymbol{\phi}_j^{3D} - \hat{\boldsymbol{\phi}}_j^{3D} \right\|_2^2$, where $\boldsymbol{\phi}_j^{3D}$ and $\hat{\boldsymbol{\phi}}_j^{3D}$ are the ground truth and estimated 3D joint locations, respectively.

In our implementation, we first train the stacked hourglass network and the 3D pose regressor separately with the heat-map loss and the 3D pose loss, respectively. Then, we train the stacked hourglass network, the residual network and the Graph CNN for mesh generation with the combined loss $\mathcal{L}_{fully}$:

$$\mathcal{L}_{fully} = \lambda_{\mathcal{H}} \mathcal{L}_{\mathcal{H}} + \lambda_{\mathcal{M}} \mathcal{L}_{\mathcal{M}} + \lambda_{\mathcal{J}} \mathcal{L}_{\mathcal{J}}, \quad (6)$$

where $\lambda_{\mathcal{H}} = 0.5$, $\lambda_{\mathcal{M}} = 1$, $\lambda_{\mathcal{J}} = 1$.

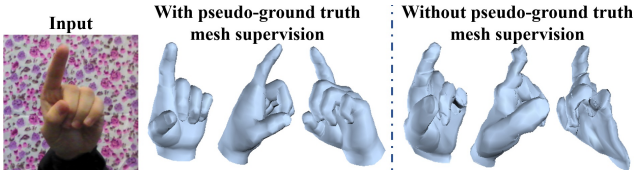

Figure 6: Impact of the pseudo-ground truth mesh supervision. Without the supervision of pseudo-ground truth mesh, the network produces very rough meshes with incorrect shape and noisy surface.

4.4. Weakly-supervised Fine-tuning

On the real-world dataset, *i.e.*, the Stereo Hand Pose Tracking Benchmark [62], there is no ground truth of 3D hand mesh. Thus, we fine-tune the networks in a weakly-supervised manner. Moreover, our model also supports the fine-tuning without the ground truth of 3D joint locations, which can further removes the burden of annotating 3D joint locations on training data and make it more applicable for large-scale real-world dataset.

Depth Map Loss. As shown in Fig. 3 (b), we leverage the reference depth map, which can be easily captured by a depth camera, as a weak supervision, and employ a differentiable renderer, similar to [22], to render the estimated 3D hand mesh to a depth map from the camera viewpoint. We use smooth L1 loss [16] for the depth map loss:

$$\mathcal{L}_{\mathcal{D}} = smooth_{L1}\left(D, \hat{D}\right),\ \hat{\mathcal{D}} = \mathcal{R}\left(\hat{\mathcal{M}}\right), \qquad (7)$$

where $\mathcal{D}$ and $\hat{\mathcal{D}}$ denote the ground truth and rendered depth maps, respectively; $\mathcal{R}(\cdot)$ is the depth rendering function; $\hat{\mathcal{M}}$ is the estimated 3D hand mesh. We set the resolution of a depth map as 32×32 px.

Pseudo-Ground Truth Mesh Loss. Training with only the depth map loss could lead to a degenerated solution, as shown in Fig. 6 (right), since the depth map loss only constrains the visible surface and is sensitive to the noise in the captured depth map. To solve this issue, inspired by [25], we create the pseudo-ground truth mesh $\tilde{\mathcal{M}}$ by testing on the real-world training data using the pretrained models and the ground truth heat-maps. The pseudo-ground truth mesh $\tilde{\mathcal{M}}$ usually has reasonable edge length and good surface smoothness, although it suffers from the relative depth error. Based on this observation, we do not apply vertex loss or normal loss, and we only adopt the edge loss $\mathcal{L}_e$ and the Laplacian loss $\mathcal{L}_l$ as the pseudo-ground truth mesh loss $\mathcal{L}_{p\mathcal{M}} = \lambda_e\mathcal{L}_e + \lambda_l\mathcal{L}_l$, where $\lambda_e = 1$, $\lambda_l = 50$, in order to preserve the edge length and surface smoothness of the mesh. As shown in Fig. 6 (middle), with the supervision of the pseudo-ground truth meshes, the network can generate meshes with correct shape and smooth surface.

In our implementation, we first fine-tune the stacked hourglass network with the heat-map loss, and then end-to-end fine-tune all networks with the combined loss $\mathcal{L}_{weakly}$:

$$\mathcal{L}_{weakly} = \lambda_{\mathcal{H}}\mathcal{L}_{\mathcal{H}} + \lambda_{\mathcal{D}}\mathcal{L}_{\mathcal{D}} + \lambda_{p\mathcal{M}}\mathcal{L}_{p\mathcal{M}}, \qquad (8)$$

where $\lambda_{\mathcal{H}} = 0.1$, $\lambda_{\mathcal{D}} = 0.1$, $\lambda_{p\mathcal{M}} = 1$. Note that Eq. 8 is the loss function for fine-tuning on the dataset without 3D pose supervision. When the ground truth of 3D joint locations is provided during training, we add the 3D pose loss $\mathcal{L}_{\mathcal{J}}$ in the loss function and set the weight $\lambda_{\mathcal{J}} = 10$.

5. Experiments

5.1. Datasets, Metrics and Implementation Details

In this work, we evaluate our method on two aspects: 3D hand mesh reconstruction and 3D hand pose estimation.

For 3D hand mesh reconstruction, we evaluate the generated 3D hand meshes on our proposed synthetic and real-world datasets, which are introduced in Section 3, since no other hand RGB image dataset contains the ground truth of 3D hand meshes. We measure the average error in Euclidean space between the corresponding vertices in each generated 3D mesh and its ground truth 3D mesh. This metric is denoted as "mesh error" in the following experiments.

For 3D hand pose estimation, we evaluate our proposed methods on two publicly available datasets: Stereo Hand Pose Tracking Benchmark (STB) [62] and the Rendered Hand Pose Dataset (RHD) [63]. STB is a real-world dataset containing 18,000 images with the ground truth of 21 3D hand joint locations and corresponding depth images. Following [63, 4, 44], we split the dataset into 15,000 training samples and 3,000 test samples. To make the joint definition consistent with our settings and RHD dataset, following [4], we move the root joint location from palm center to wrist. RHD is a synthetic dataset containing 41,258 training images and 2,728 testing images. This dataset is challenging due to the large variations in viewpoints and the low image resolution. We evaluate the performance of 3D hand pose estimation with three metrics: (i) Pose error: the average error in Euclidean space between the estimated 3D joints and the ground truth joints; (ii) 3D PCK: the percentage of correct keypoints of which the Euclidean error distance is below a threshold; (iii) AUC: the area under the curve on PCK for different error thresholds.

We implement our method within the PyTorch framework. The networks are trained using the RMSprop optimizer [50] with mini-batches of size 32. The learning rate is set as 10^{-3} when pretraining on our synthetic dataset, and is set as 10^{-4} when fine-tuning on RHD [63] and STB [62]. The input image is resized to 256×256 px. Following the same condition used in [63, 4, 44], we assume that the global hand scale and the absolute depth of root joint are provided at test time. The global hand scale is set as the length of the bone between MCP and PIP joints of the middle finger.

Error (mm)	−Normal	−Edge	−Laplacian	−3D Pose	Full
Mesh error	8.34	9.09	8.63	9.04	**7.95**
Pose error	8.30	9.06	8.55	9.24	**8.03**

Table 1: Ablation study by eliminating different loss terms from our fully-supervised training loss in Eq. 6, respectively. We report the average mesh and pose errors evaluated on the validation set of our synthetic dataset.

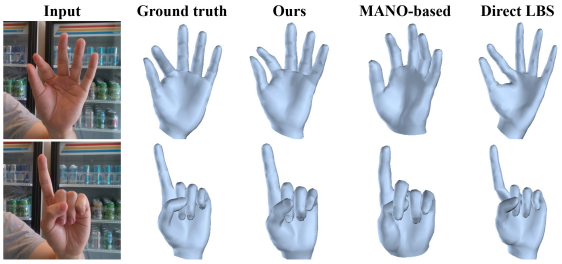

Figure 7: Qualitative comparisons of the meshes generated by our method and other methods. The meshes generated by the MANO-based method usually exhibit inaccurate shape and pose. The meshes generated by the direct Linear Blend Skinning (LBS) method suffer from serious artifacts. Examples are taken from our real-world dataset.

Mesh error (mm)	MANO-based	Direct LBS	Ours
Our synthetic dataset	12.12	10.32	**8.01**
Our real-world dataset	20.86	13.33	**12.72**

Table 2: Average mesh errors tested on the validation set of our synthetic dataset and our real-world dataset. We compare our method with two baseline methods. Note that the mesh errors in this table are measured on the aligned mesh defined by MANO [42] for fair comparison.

5.2. Ablation Study of Loss Terms

We first evaluate the impact of different losses used in the fully-supervised training (Eq. 6) on the performance of mesh reconstruction and pose estimation. We conduct this experiment on our synthetic dataset. As presented in Table 1, the model trained with the full loss achieves the best performance in both mesh reconstruction and pose estimation, which indicates that all the losses have contributions to producing accurate 3D hand mesh as well as 3D hand joint locations.

5.3. Evaluation of 3D Hand Mesh Reconstruction

We demonstrate the advantages of our proposed Graph CNN-based 3D hand mesh reconstruction method by comparing it with two baseline methods: direct Linear Blend Skinning (LBS) method and MANO-based method.

Direct LBS. We train the network to directly regress 3D hand joint locations from the heat-maps and the image features, which is similar to the network architecture proposed in [4]. We generate the 3D hand mesh from only the estimated 3D hand joint locations by applying inverse kinematics and LBS with the predefined mesh model and skinning weights (see the supplementary for details). As shown in Table 2, the average mesh error of direct LBS method is worse than our method on both our synthetic dataset and our real-world dataset, since the LBS model for mesh generation is predefined and cannot be adapt to hands with different shapes. As can be seen in Fig. 7, the hand meshes generated by direct LBS method have unrealistic deformation at joints and suffer from serious inherent artifacts.

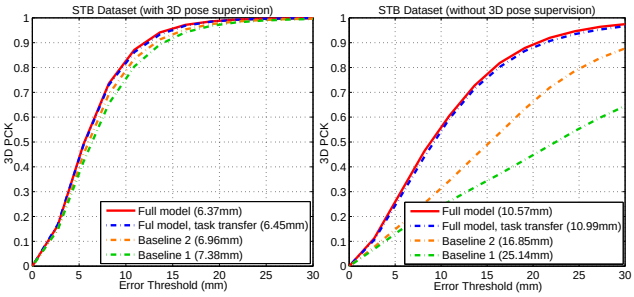

Figure 8: Self-comparisons of 3D hand pose estimation on STB dataset [62]. **Left**: 3D PCK of the model fine-tuned with 3D hand pose supervision. **Right**: 3D PCK of the model fine-tuned without 3D hand pose supervision. The average pose errors are shown in parentheses.

Method	Pipeline	Depth map loss
Baseline 1	im→hm+feat→pose	✗
Baseline 2	im→hm+feat→mesh→pose	✗
Full model	im→hm+feat→mesh→pose	✓

Table 3: Differences between the baseline methods for 3D hand pose estimation and our full model.

MANO-based Method. We also implement a MANO [42] based method that regresses hand shape and pose parameters from the latent image features using three fully-connected layers. Then, the 3D hand mesh is generated from the estimated shape and pose parameters using MANO hand model [42] (see the supplementary for details). The networks are trained in fully-supervised manner using the same loss functions as Eq. 6 on our synthetic dataset. For fair comparison, we align our hand mesh with the MANO hand mesh, and compute mesh error on the aligned mesh. As shown in Table 2 and Fig. 7, the MANO-based method exhibits inferior performance on mesh reconstruction compared with our method. The reason may lie in the difficulty of regressing accurate MANO parameters directly from image features and the limited representation power of the MANO hand model with linear bases.

5.4. Evaluation of 3D Hand Pose Estimation

We also evaluate our approach on the task of 3D hand pose estimation.

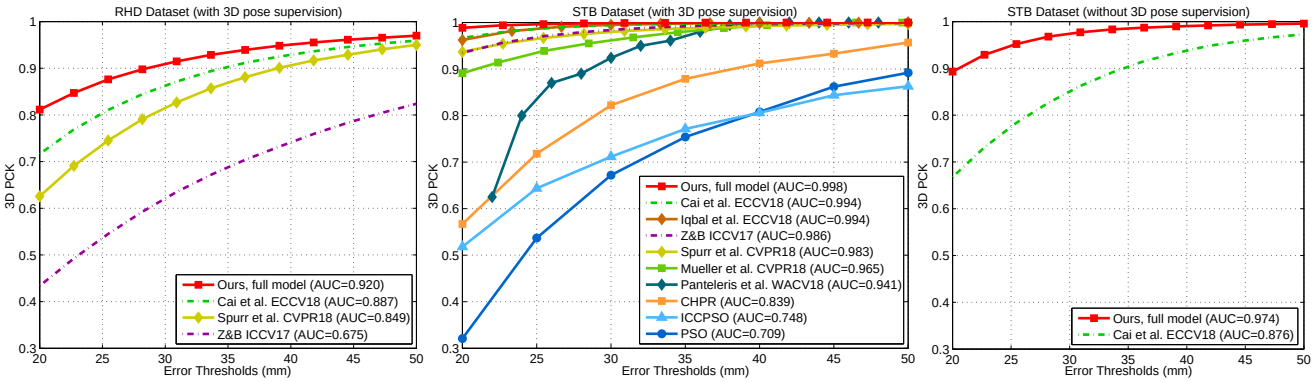

Figure 9: Comparisons with state-of-the-art methods on RHD [63] and STB [62] dataset. **Left**: 3D PCK on RHD dataset [63] with 3D hand pose supervision. **Middle**: 3D PCK on STB dataset [62] with 3D hand pose supervision. **Right**: 3D PCK on STB dataset [62] without 3D hand pose supervision. The AUC values are shown in parentheses.

Self-comparisons. We conduct self-comparisons on STB dataset [62] by fine-tuning the networks pretrained on our synthetic dataset in a weakly-supervised manner, as described in Section 4.4. In Table 3, we compare our proposed weakly-supervised method (**Full model**) with two baselines: (i) **Baseline 1**: directly regressing 3D hand joint locations from the heat-maps and the feature maps without using the depth map loss during training; (ii) **Baseline 2**: regressing 3D hand joint locations from the estimated 3D hand mesh without using the depth map loss during training. As presented in Fig. 8, the estimation accuracy of Baseline 2 is superior to that of Baseline 1, which indicates that our proposed 3D hand mesh reconstruction network is beneficial to 3D hand pose estimation. Furthermore, the estimation accuracy of our full model is superior to that of Baseline 2, especially when fine-tuning without 3D hand pose supervision, which validates the effectiveness of introducing the depth map loss as a weak supervision.

In addition, to explore a more efficient way for 3D hand pose estimation without mesh generation, we directly regress the 3D hand joint locations from the latent feature extracted by our full model instead of regressing them from the 3D hand mesh (see the supplementary for details). This task transfer method is denoted as "**Full model, task transfer**" in Fig. 8. Although this method has the same pipeline as that of Baseline 1, the estimation accuracy of this task transfer method is better than that of Baseline 1 and is only a little bit worse than that of our full model, which indicates that the latent feature extracted by our full model is more discriminative and is easier to regress accurate 3D hand pose than the latent feature extracted by Baseline 1.

Comparisons with State-of-the-arts. We compare our method with state-of-the-art 3D hand pose estimation methods on RHD [63] and STB [62] datasets. The PCK curves over different error thresholds are presented in Fig. 9. On RHD dataset, as shown in Fig. 9 (left), our method outperforms the three state-of-the-art methods [63, 44, 4] over all the error thresholds on this dataset. On STB dataset, when the 3D hand pose ground truth is given during training, we compare our methods with seven state-of-the-art methods [62, 63, 36, 44, 32, 4, 19], and our method outperforms these methods over most of the error thresholds, as shown in Fig. 9 (middle). We also experiment with the situation when 3D hand pose ground truth is unknown during training on STB dataset, and compare our method with the weakly-supervised method proposed by Cai *et al*. [4], both of which adopt reference depth maps as a weak supervision. As shown in Fig. 9 (right), our 3D mesh-based method outperforms Cai *et al*. [4] by a large margin.

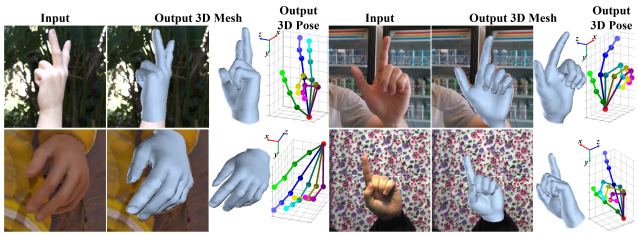

Figure 10: Qualitative results for our synthetic dataset (top left), our real-world dataset (top right), RHD dataset [63] (bottom left), and STB dataset [62] (bottom right).

5.5. Runtime and Qualitative Results

Runtime. We evaluate the runtime of our method on one Nvidia GTX 1080 GPU. The runtime of our full model outputting both 3D hand mesh and 3D hand pose is 19.9ms on average, including 12.6ms for the stacked hourglass network forward propagation, 4.7ms for the residual network and Graph CNN forward propagation, and 2.6ms for the forward propagation of the pose regressor. Thus, our method can run in real-time on GPU at over 50fps.

Qualitative Results. Some qualitative results of 3D hand mesh reconstruction and 3D hand pose estimation for our synthetic dataset, our real-world dataset, RHD [63], and

STB [62] datasets are shown in Fig. 10. More qualitative results are presented in the supplementary.

6. Conclusion

In this paper we have tackled the challenging task of 3D hand shape and pose estimation from a single RGB image. We have developed a Graph CNN-based model to reconstruct a full 3D mesh of hand surface from an input RGB image. To train the model, we have created a large-scale synthetic RGB image dataset with ground truth annotations of both 3D joint locations and 3D hand meshes, on which we train our model in a fully-supervised manner. To fine-tune our model on real-world datasets without 3D ground truth, we render the generated 3D mesh to a depth map and leverage the observed depth map as a weak supervision. Experiments on our proposed new datasets and two public datasets show that our method can recover accurate 3D hand mesh and 3D joint locations in real-time.

In future work, we will use Mocap data to create a larger 3D hand pose and shape dataset. We will also consider the cases of hand-object and hand-hand interactions in order to make the hand pose and shape estimation more robust.

Acknowledgment: This work is in part supported by MoE Tier-2 Grant (2016-T2-2-065) of Singapore. This work is also supported in part by start-up grants from University at Buffalo and a gift grant from Snap Inc.

References

[1] Autodesk. Arnold renderer. `https://www.arnoldrenderer.com`, 2018.

[2] Autodesk. Maya. `https://www.autodesk.com.sg/products/maya`, 2018.

[3] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *ECCV*, 2016.

[4] Y. Cai, L. Ge, J. Cai, and J. Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *ECCV*, 2018.

[5] F. R. Chung and F. C. Graham. *Spectral graph theory*, volume 92. American Mathematical Society, 1997.

[6] M. de La Gorce, D. J. Fleet, and N. Paragios. Model-based 3d hand pose estimation from monocular video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1793–1805, 2011.

[7] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

[8] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11), 2007.

[9] Flickr. Flickr. `https://www.flickr.com/`, 2018.

[10] L. Ge, Y. Cai, J. Weng, and J. Yuan. Hand pointnet: 3d hand pose estimation using point sets. In *CVPR*, 2018.

[11] L. Ge, H. Liang, J. Yuan, and D. Thalmann. Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs. In *CVPR*, 2016.

[12] L. Ge, H. Liang, J. Yuan, and D. Thalmann. 3D convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *CVPR*, 2017.

[13] L. Ge, H. Liang, J. Yuan, and D. Thalmann. Real-time 3D hand pose estimation with 3d convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[14] L. Ge, H. Liang, J. Yuan, and D. Thalmann. Robust 3d hand pose estimation from single depth images using multi-view cnns. *IEEE Transactions on Image Processing*, 27(9):4422–4436, 2018.

[15] L. Ge, Z. Ren, and J. Yuan. Point-to-point regression pointnet for 3d hand pose estimation. In *ECCV*, 2018.

[16] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] Intel. Intel realsense. `https://realsense.intel.com/`, 2018.

[19] U. Iqbal, P. Molchanov, T. Breuel, J. Gall, and J. Kautz. Hand pose estimation via latent 2.5 d heatmap regression. In *ECCV*, 2018.

[20] D. Joseph Tan, T. Cashman, J. Taylor, A. Fitzgibbon, D. Tarlow, S. Khamis, S. Izadi, and J. Shotton. Fits like a glove: Rapid and reliable hand shape personalization. In *CVPR*, 2016.

[21] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018.

[22] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *CVPR*, 2018.

[23] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *CVPR*, 2015.

[24] C. Lassner, J. Romero, M. Kiefel, F. Bogo, M. J. Black, and P. V. Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *CVPR*, 2017.

[25] Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2017.

[26] H. Liang, J. Yuan, J. Lee, L. Ge, and D. Thalmann. Hough forest with optimized leaves for global hand pose estimation with arbitrary postures. *IEEE Transactions on Cybernetics*, 49(2):527–541, 2019.

[27] J. Lin, Y. Wu, and T. S. Huang. Modeling the constraints of human hand motion. In *Proceedings of the Workshop on Human Motion*, 2000.

[28] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[29] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):248, 2015.

[30] A. Makris and A. Argyros. Model-based 3d hand tracking with on-line hand shape adaptation. *BMVC*, 2015.

[31] J. Malik, A. Elhayek, F. Nunnari, K. Varanasi, K. Tamaddon, A. Heloir, and D. Stricker. Deephps: End-to-end estimation

of 3d hand pose and shape by learning from synthetic depth. In *3DV*, 2018.

[32] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt. GANerated hands for real-time 3d hand tracking from monocular RGB. In *CVPR*, 2018.

[33] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt. Real-time hand tracking under occlusion from an egocentric RGB-D sensor. In *ICCV*, 2017.

[34] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[35] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *BMVC*, 2011.

[36] P. Panteleris, I. Oikonomidis, and A. Argyros. Using a single rgb frame for real time 3d hand pose estimation in the wild. In *WACV*, 2018.

[37] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis. Learning to estimate 3d human pose and shape from a single color image. *CVPR*, 2018.

[38] M. Rad, M. Oberweger, and V. Lepetit. Domain transfer for 3d pose estimation from color images without manual annotations. In *ACCV*, 2018.

[39] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. Generating 3d faces using convolutional mesh autoencoders. *ECCV*, 2018.

[40] J. M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. In *ECCV*, 1994.

[41] E. Remelli, A. Tkach, A. Tagliasacchi, and M. Pauly. Low-dimensionality calibration through local anisotropic scaling for robust hand model personalization. In *ICCV*, 2017.

[42] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (TOG)*, 36(6):245, 2017.

[43] T. Simon, H. Joo, I. A. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[44] A. Spurr, J. Song, S. Park, and O. Hilliges. Cross-modal deep variational hand pose estimation. In *CVPR*, 2018.

[45] S. Sridhar, F. Mueller, M. Zollhöfer, D. Casas, A. Oulasvirta, and C. Theobalt. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *ECCV*, 2016.

[46] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *ICCV*, 2013.

[47] B. Stenger, A. Thayananthan, P. H. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1372–1384, 2006.

[48] J. Tan, I. Budvytis, and R. Cipolla. Indirect deep structured learning for 3d human body shape and pose prediction. In *BMVC*, 2017.

[49] J. Taylor, R. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Hertzmann, and A. Fitzgibbon. User-specific hand modeling from monocular depth sequences. In *CVPR*, 2014.

[50] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 2012.

[51] A. Tkach, A. Tagliasacchi, E. Remelli, M. Pauly, and A. Fitzgibbon. Online generative model personalization for hand tracking. *ACM Transactions on Graphics (TOG)*, 36(6):243, 2017.

[52] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):169, 2014.

[53] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki. Self-supervised learning of motion capture. In *NIPS*, 2017.

[54] G. Varol, D. Ceylan, B. Russell, J. Yang, E. Yumer, I. Laptev, and C. Schmid. Bodynet: Volumetric inference of 3d human body shapes. *ECCV*, 2018.

[55] N. Verma, E. Boyer, and J. Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *CVPR*, 2018.

[56] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018.

[57] Y. Wu and T. S. Huang. Hand modeling, analysis and recognition. *IEEE Signal Processing Magazine*, 18(3):51–60, 2001.

[58] Y. Wu, J. Lin, and T. S. Huang. Analyzing and capturing articulated hand motion in image sequences. *IEEE transactions on pattern analysis and machine intelligence*, 27(12):1910–1922, 2005.

[59] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:1801.07455*, 2018.

[60] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[61] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Yong Chang, K. Mu Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, et al. Depth-based 3d hand pose estimation: From current achievements to future goals. In *CVPR*, 2018.

[62] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu, and Q. Yang. 3D hand pose tracking and estimation using stereo matching. *arXiv preprint arXiv:1610.07214*, 2016.

[63] C. Zimmermann and T. Brox. Learning to estimate 3d hand pose from single RGB images. In *ICCV*, 2017.

Supplementary

A. Qualitative Results

We present more qualitative results of 3D hand mesh reconstruction and 3D hand pose estimation for our synthetic dataset, our real-world dataset, STB dataset [62], RHD dataset [63], and Dexter+Object dataset [45], as shown in Fig. 11. Please see the supplementary video for more qualitative results on continuous sequences.

B. Details of Baseline Methods for 3D Hand Mesh Reconstruction

In Section 5.3 of our main paper, we compare our proposed method with two baseline methods for 3D hand mesh reconstruction: direct Linear Blend Skinning (LBS) method and MANO-based method. Here, we describe more details of these two baseline methods, as illustrated in Fig. 12.

In the direct LBS method, we train the network to regress 3D hand joint locations from the heat-maps and the image features with heat-map loss and 3D pose loss. As illustrated in Fig. 12 (b), the latent feature extracted from the input image is mapped to 3D hand joint locations through a multi-layer perceptron (MLP) network with three fully-connected layers. Then, we apply inverse kinematics (IK) to compute the transformation matrix of each hand joint from the the estimated 3D hand joint locations. The 3D hand mesh is generated by applying LBS with the predefined hand model and skinning weights. In this method, the 3D hand mesh is only determined by the estimated 3D hand joint locations, thus it cannot be adapted to various hand shapes. In addition, the IK often suffers from singularity and multiple solutions, which makes the solutions to transformation matrices unreliable. Experimental results in Figure 7 and Table 2 of our main paper have shown the limitations of this direct LBS method.

In the MANO-based method, we train the network to regress hand shape and pose parameters of the MANO hand model [42]. As illustrated in Fig. 12 (c), the latent feature extracted from the input image is mapped to hand shape and pose parameters θ, β through an MLP network with three fully-connected layers. Then, the 3D hand mesh is generated from the regressed parameters θ, β using the MANO hand model [42]. Note that the MANO mesh generation module is differentiable and is involved in the network training. The networks are trained with heat-map loss, mesh loss and 3D pose loss, which are the same as our method. Since the MANO hand model is fixed during training and is essentially LBS with blend shapes [42], the representation power of this method is limited. Experimental results in Figure 7 and Table 2 of our main paper have shown the limitations of this MANO-based method.

C. Details of the Task Transfer Method

In Section 5.4 of our main paper, we implement an alternative method ("full model, task transfer") for 3D hand pose estimation by transferring our full model trained for 3D hand mesh reconstruction to the task of 3D hand pose estimation. Here, we describe more details of our task transfer method. As illustrated in Fig. 13, we directly regress the 3D hand joint locations from the latent feature extracted by our full model using an MLP network with three fully-connected layers. We first train the MLP network with 3D pose loss on our synthetic dataset. When experimenting on STB dataset [62] with 3D pose supervision, we fine-tune the MLP network with 3D pose loss. When experimenting on STB dataset [62] without 3D pose supervision, we directly use the MLP network pretrained on our synthetic dataset. Experimental results in Figure 8 of our main paper show that our task transfer method is better than the baseline method which is only trained for 3D hand pose estimation, even though these two methods have the same pipeline. This indicates that the latent feature extracted by our full model is more discriminative and is easier to regress accurate 3D hand pose since our full model is trained with the dense supervision of the 3D hand mesh that contains richer information than the 3D hand pose. In addition, although the estimation accuracy of our task transfer method is a little bit worse than that of our full model, our task transfer method is faster than our full model, since it does not generate 3D hand mesh. The runtime of our task transfer method is 15.1ms, while the runtime of our full model which estimate 3D hand pose from hand mesh is 19.9ms. Thus, in applications that only require 3D hand pose estimation but not 3D hand shape estimation, we can choose to use this task transfer method, which can maintain a comparable accuracy as our full model while runs at faster speed.

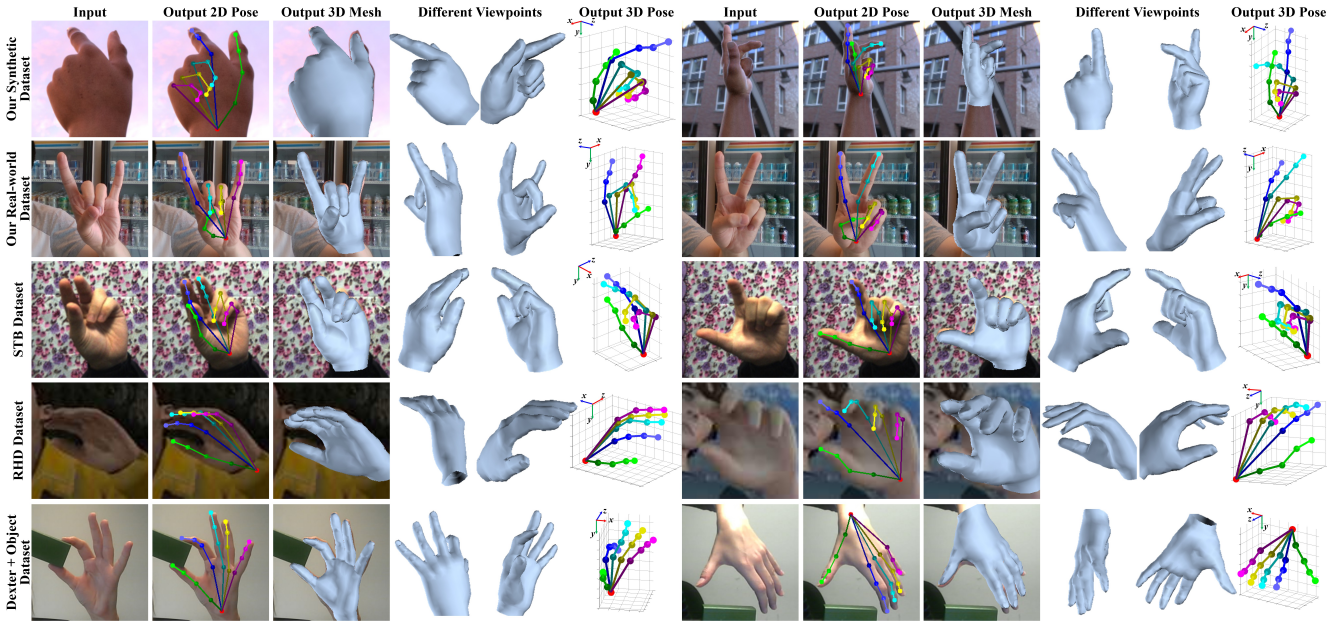

Figure 11: Qualitative results for our synthetic dataset (the first row), our real-world dataset (the second row), STB dataset [62] (the third row), RHD dataset [63] (the fourth row), and Dexter+Object dataset [45] (the last row).

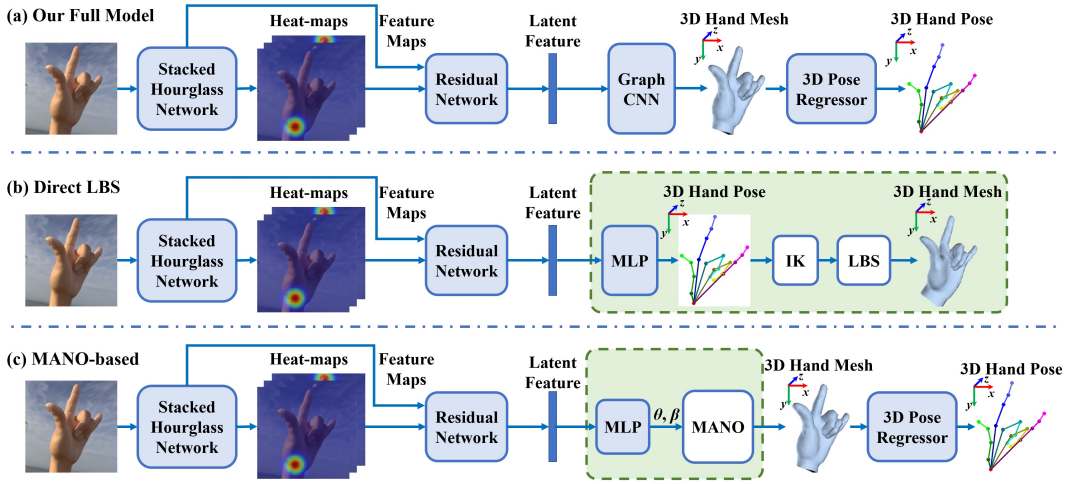

Figure 12: Pipelines of our proposed method and two baseline methods: direct LBS method and MANO-based method. The differences between the two baseline methods and our proposed method are highlighted in the green dashed line box.

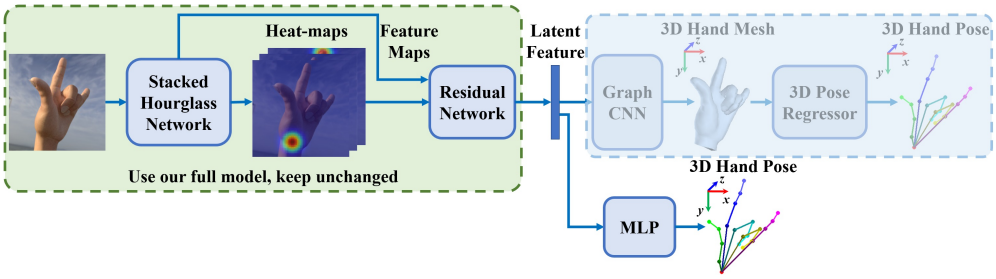

Figure 13: Illustration of our "full model, task transfer" method. We transfer our full model trained for 3D hand mesh reconstruction to the task of 3D hand pose estimation. Note that when training for the task of 3D hand pose estimation, the stacked hourglass network and the residual network are keep unchanged with our full model which is fully trained for the task of 3D hand mesh reconstruction.