Dual Grid Net: Hand Mesh Vertex Regression from Single Depth Maps

Chengde Wan[1], Thomas Probst[2], Luc Van Gool[2], Angela Yao[3],

[1] Facebook Reality Labs
[2] Computer Vision Laboratory, ETH Zürich
[3] National University of Singapore

Abstract. We aim to recover the dense 3D surface of the hand from depth maps and propose a network that can predict mesh vertices, transformation matrices for every joint and joint coordinates in a single forward pass. Use fully convolutional architectures, we first map depth image features to the mesh grid and then regress the mesh coordinates into real world 3D coordinates. The final mesh is found by sampling from the mesh grid refit in closed-form based on an articulated template mesh. When trained with supervision from sparse key-points, our accuracy is comparable with state-of-the-art on the NYU dataset for key point localization, all while recovering mesh vertices and dense correspondences. Under multi-view settings for training, our framework can also learn through self-supervision by minimizing a set of data-fitting terms and kinematic priors. Our approach is competitive with strongly supervised methods and showcases the potential for self-supervision in dense mesh estimation.

1 Introduction

We consider the problem of estimating 3D shape and pose of articulated objects from single depth images. Specifically, we want to estimate the position of surface mesh vertices of the human hand model. Unlike skeleton joints, dense mesh vertices encode both pose and shape of the hand and enable a much wider range of virtual and mixed reality applications. For example, one can directly put the virtual hand in a VR game, or overlay a user's hand surface with another texture map in mixed reality. Furthermore, manipulation of virtual objects can naturally be modelled through interaction of dense surface representations.

Estimating mesh vertices is significantly more challenging than estimating skeleton joints. First, the scale of the problem increases by several magnitudes. To reasonably represent a human hand, one needs thousands of mesh vertices, as opposed to tens of joint positions and angles. Secondly, getting accurate 3D ground truth for the thousands of vertices from real-world data is extremely difficult even though having large amounts of labelled training data is crucial for data-driven learning based methods.

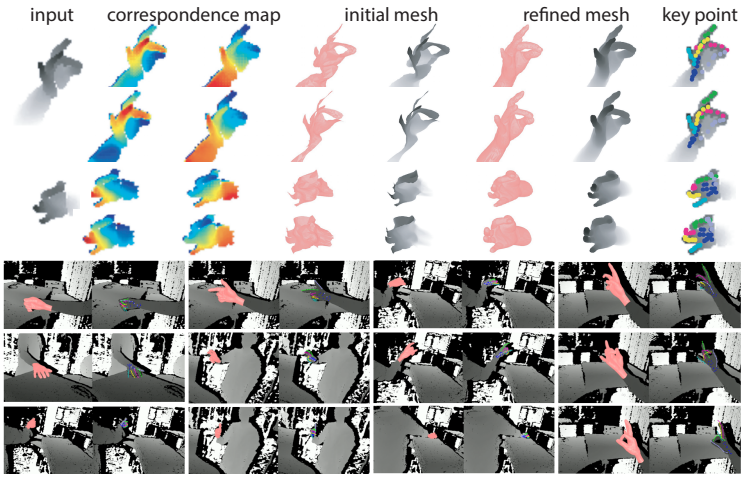


Fig. 1: Upper rows: qualitative results on NYU [55]. In each group, upper rows are results supervised with key-point annotations and lower rows with self-supervision. We visualize the correspondence map with each mesh coordinate, the rendered shading and depth map of the initial estimated mesh model and refined ones, as well as key-points. Bottom rows: qualitative results from real-world data with multiple users and view points showing the estimated mesh and corresponding keypoints.

The most recent works that estimate mesh vertices leverage deep methods such as VoxelNet [57], graph convolutions [37, 13], or parametric models [5, 68, 23]. These approaches have made significant advances for hand pose estimation but are not without drawbacks. They tend to be restricted to fixed mesh topologies, have a very large number of network parameters, are difficult to train, and or are limited in spatial resolution. The use of parametric models such as SMPL [22] and MANO [38] has made 3D mesh estimation highly accessible. The models are highly compact; for example, MANO has 19[16] dimensions for each hand. But by directly estimating shape parameters and joint angles of the mesh, these parametric approaches may not capture finer spatial details. They are also sensitive to perturbations, since small offsets from a single dimension of an estimate easily propagates to many mesh vertices.

We were motivated to develop a method that disentangles hand pose from shape estimation and is able to explicitly enforce estimated pose aligned with pre-calibrated hand shapes when available. Since both captured inputs and meshes are inherently surfaces, it is natural to consider them as a 2D embedding in a 3D Euclidean space. To this end, we propose solving mesh vertex regression with a fully 2D convolutional architecture that learns the extrinsic geometric properties of 3D inputs as well as intrinsics of the mesh model. Our approach is easy to train, highly efficient, and flexible enough to handle different mesh topologies and templates. Moreover, we can also capture very fine spatial detailing through per-

pixel correspondences to a mesh model, thereby allowing finer spatial resolution and for better alignment between the mesh model and depth observations.

At the core of our method are two 2D fully convolutional networks (FCNs), applied to the image and mesh estimates consecutively (see Fig. 2). Linking the FCNs is a 2D embedding that propagates gradients directly from the irregular representation of a mesh to the regular and ordered representation of an image. To refine the estimated mesh, we solve for a similarity transform with singular value decomposition (SVD) to a template hand mesh model. We then re-pose the template mesh based on the transform to yield a denoised mesh surface together with key points. Since SVD has closed form solutions and is a differentiable operator, one can also place supervision on the estimated key points.

We first pre-trained our network on a synthetic dataset. Afterwards, the network can be fine tuned to real-world data by either feeding sparse key-point annotations or by directly minimizing the reconstruction error between the mesh estimation and observations. For the latter case, we propose a self-supervision scheme that minimizes a geometric model-fitting energy as a training loss. The model's accuracy steadily improves with increasing amounts of data seen, even without any human-provided labels. Finally, since correspondences between observed hand pixels and the mesh are estimated in a differentiable way, we can optimize the correspondences jointly with the disparity between the correspondence pairs during model-fitting. This differs from and complements standard ICP optimization methods. Such a self-supervision scheme greatly improves the accuracy trained by synthetic data only. To further resolve the self-occlusion, a multi-view consistency term can be optionally added when a multi-view camera setup is available. In the multi-view camera setup, the proposed self-supervision method can achieve competitive accuracy to supervised state-of-the-art.

Our contributions can be summarized as follows,

- We propose a novel fully convolutional network architecture for regressing thousands of mesh vertices in an end-to-end manner.
- A self-learning scheme is proposed for training the network; without any human labels, our network achieves competitive results when compared to fully supervised state-of-the-art. Such a learning approach offers a new and accurate way of annotating real-world data and thereby solves one of the key difficulties in making progress for hand pose estimation.
- Our method bridges a gap between data-driven discriminative methods and optimization-based model-fitting and benefits from both: accuracy that improves with the amount of data shown, while not needing human annotations.

2 Related Works

Hand pose estimation. Deep learning has significantly advanced state-of-the-art for hand pose estimation. The general trend has been to develop deeper and more complex network architectures [7, 27, 8, 14, 24, 11, 61, 63]. Such progress has hinged on having large amounts of annotated data [55, 67, 43]. Obtaining accurate annotations, even for simple 3D joint coordinates, is extremely difficult and

time consuming. Annotations generated by manually initializing trackers [55, 28] require carefully designed interfaces for 3D annotation and there is often large discrepancies between human annotators [48]. Motion-capture rigs [43] and auxiliary sensors [67] are fully automatic but have limited deployment environments. To mitigate the lack of annotations, semi-supervised approaches [60, 6, 33] and approaches coupling real and synthetic data [42, 32, 36] have also been proposed.

An alternative line of work [53, 35, 49, 40, 51, 18, 46, 54, 25] estimates pose by minimizing a model-fitting error. Model-fitting needs little to no human labels, but the accuracy is heavily dependent on the careful design of the energy function. A recent trend bridges data-driven and model-fitting approaches [56, 10, 13, 59] by using a differentiable renderer and incorporating the model-fitting error as a part of the training loss. Our work continues in this trend, but differs from previous methods in two key respects. First, we re-parameterize the mesh with a 2D embedding, which allows us to use a 2D fully convolutional network architecture. Secondly, we apply self-supervision on both the image grid and the mesh grid, leading to efficient gradient flows during back-propagation.

Human mesh model recovery from single image. Data-driven methods have greatly advanced the 3D reconstruction of shape and pose of the full body [52, 62, 3, 30, 50, 31, 56, 19, 57, 39, 65], face [17, 21, 66, 37] and hands [54, 17, 23, 68, 5, 13, 16]. Earlier works focused on landmark detection[3], segmentation[54], and finding correspondences [52, 62, 17, 66, 25], and performed a model-based optimization to fit the mesh in a subsequent step. Recently, trends have shifted to end-to-end learning of the mesh with neural networks. Several works [30, 19, 31, 56, 68, 5, 23, 65, 16] favour parametric models like SMPL [22] and MANO [38].

Various encoder-decoder frameworks have also been used, applying graph convolution to mesh vertices [37, 13], VoxelNet to 3D occupancy grids[57], and fully connected and transposed convolutions to silhouettes [50] and texture and mesh vertices [21]. Unlike these works, our approach is based on correspondence estimation. Yet we also differ from other correspondence-based methods [62, 52, 1, 17, 66] in that we directly estimate mesh vertices with a single forward pass.

3D Network Architectures. It is highly intuitive to parameterize 3D inputs and outputs as an occupancy grid or distance field and use a 3D architecture [12, 57, 24]. Networks such as VoxelNet however are parameter heavy and severely limited in spatial resolution. PointNet [34] is a light-weight alternative and while it can interpret 3D inputs a set of un-ordered points, it also largely ignores spatial contexts which may be important downstream.

Since captured 3D inputs are inherently object surfaces, it is natural to consider them as 2D embeddings in 3D space. Several works [9, 20, 37] have modeled mesh surfaces as a graph and have applied graph network architectures to capture intrinsic and extrinsic geometric properties of the mesh. Our method also works on the hand surface, but it is a simpler and more flexible network architecture which is easier to train. Our method most resembles [2, 47] by mapping high dimension data to a 2D grid. However, instead of just working on points from the depth map, we use dual grids, enabling the mapping of heterogeneous data from Euclidean space to mesh surfaces and vice versa.

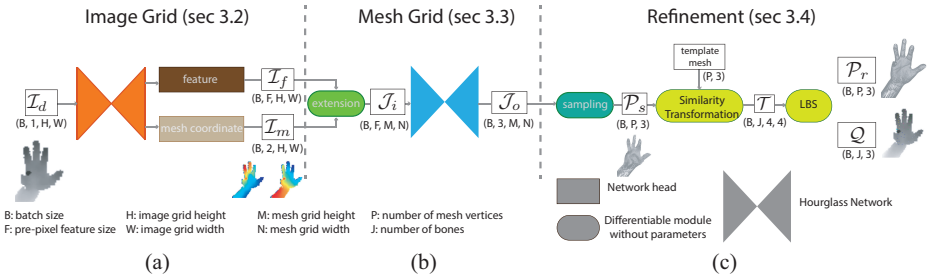


Fig. 2: **System Framework.** Starting from a depth map of the segmented hand as input, we estimate a dense correspondence map to the mesh model for every point on the image grid (Sec. 3.2). This correspondence maps features from the image grid to the mesh grid and allows us to recover the 3D coordinates of all the mesh vertices (Sec. 3.3) on the mesh grid. Finally, coordinates are refined by skinning a template mesh model with respect to the recovered vertices (Sec. 3.4).

3 Dual Grid Net

Our Dual Grid Net (DGN) is an efficient fully convolutional network architecture for mesh vertex estimation. At its core are consecutive 2D convolutions on two grids – an image grid and a mesh grid – where features from one grid can be mapped to another differentiably. We assume we are provided a canonical hand mesh model which is generic and applicable to all users' hands. In a given depth map, every pixel on the hand's surface has a correspondence to the mesh surface. Finding these correspondences is equivalent to mapping pixel coordinates from the image grid to the mesh grid (Sec. 3.1). Armed with a dense correspondence (Sec. 3.2) we map features from the image grid to the mesh grid and recover the 3D coordinates of all the mesh vertices (Sec. 3.3). We further refine these coordinates by skinning a template mesh model with respect to the recovered mesh vertices (Sec. 3.4). The entire process is illustrated in Fig. 2.

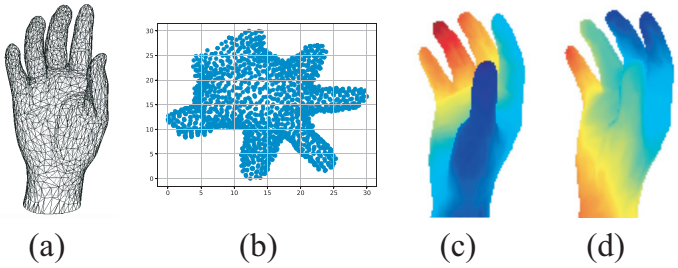


Fig. 3: (a) Triangular mesh model used in this work; (b) 2D MDS embedding of the mesh vertices; (c, d) mesh coordinates on mesh surface corresponding to 2D MDS embedding.

3.1 Mesh model

We use a triangle mesh model (see Fig. 3(a)) with 1721 vertices. Every point on the mesh surface has a pair of *"intrinsic"* coordinates which depend only on its position on the mesh and is therefore invariant to hand pose, shape, or view point. In addition, we consider *"extrinsic"* properties of points on the mesh surface such as texture, colour, or its 3D coordinates in the camera. Both the intrinsics and extrinsics of each mesh vertex can be approximated via linear interpolation of neighbouring points on the mesh surface.

A common way to parameterize mesh coordinates is via UV maps [1]. We follow a similar approach and use multidimensional scaling (MDS) [4] to parameterize the mesh. For any two points on a mesh surface, MDS aims to keep their Cartesian distance *w.r.t.* the mesh coordinates to be as close as possible to the geodesic distance on the mesh surface. We set the dimension of mesh coordinates (a.k.a. the intrinsic coordinates) to 2, to allow for 2D convolutions on the mesh grid. The learned MDS embedding used in this work is shown in Fig. 3(b), and the corresponding mesh coordinates projected onto the 3D mesh surface in Fig. 3(c) and (d) respectively.

3.2 Mesh Coordinate Estimation

Similar to [1], we first estimate the 2D mesh coordinates for all pixels from the hand region. We adopt an hourglass network [26] (see Fig. 2) as the backbone architecture and apply it in two heads. The first head estimates the 2D mesh coordinates $\mathcal{I}_m$ for all depth pixels while the second head estimates a generic feature map $\mathcal{I}_f$ which is later mapped to the mesh grid. Unlike [17], which performs classification followed by residual regression, we adopt a direct regression approach, which we find achieves sufficient accuracy.

Previous works [13, 5, 68, 23] encoded image inputs as a fixed-size latent vector. Our approach, by using dense mesh coordinates, has two major advantages. Firstly, it allows us to use an FCN architecture. This important difference means we can maintain spatial resolution but also has advantages of efficiency and translational invariance. It is also much easier for learning, since one can directly apply pixel-wise supervision on both image grid and mesh grid. Secondly, the estimated mesh coordinates establishes a dense correspondence map between the captured hand surface and the mesh surface. The correspondence map, as we will show in Sec. 4.1, allows us to directly embed a lifting energy [18], which is beneficial to minimizing the model-fitting error in a self-supervised setting.

3.3 Mapping from image grid to mesh grid

In this section, we describe the recovery of all mesh vertices, including occluded ones, from the estimated per-pixel mesh coordinates and features on the image grid. Based on the estimated mesh coordinates, feature maps computed from the depth image can be mapped from the image grid to the mesh grid. Similar to [2], we call this process *extension* (see Fig. 4).

More specifically, for any pixel p belonging to the hand surface, we can regress its coordinate on the mesh grid $m = (m_x, m_y) \in \mathcal{R}^2$ as well as its corresponding feature $f \in \mathcal{R}^d$ as obtained by the feature head as described in Sec. 3.2. f is propagated to the mesh grid via soft assignment to the neighbours of m:

$$f \underset{\text{sampling}}{\overset{\text{propagation}}{\rightleftharpoons}} \sum_{n \in \Omega(m)} w_n \cdot f_n. \tag{1}$$

f is propagated to the grid point n with a weighting determined by the softmax of its distance to m as follows, where $\sigma = 0.5$:

$$w_n = \frac{e^{-\sigma(n-m)^2}}{\sum_l e^{-\sigma(l-m)^2}}. \tag{2}$$

We adopt a second hourglass network on the mesh grid to recover all mesh vertices. Given that every mesh vertex is associated with a fixed mesh coordinate, the output features of hourglass network is aggregated according to their mesh coordinates of vertices. In turn, this process is named as *sampling* (see Fig. 4).

Note that propagated features will only partially occupy the mesh grid due to occlusions. However, the sampling process requires features from all over the mesh grid. This resembles an image in-painting process and we leverage the encoder-decoder structure of the hourglass to utilize both global and local context when filling in these values.

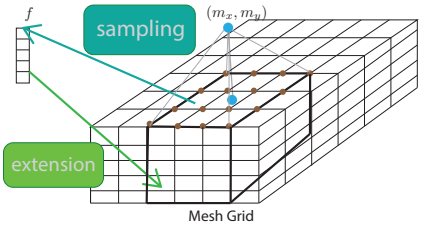


Fig. 4: Illustrations of the extension and sampling process, where $f \in \mathcal{R}^f$ is the mapped feature and $(m_x, m_y) \in \mathcal{R}^2$ is its corresponding coordinate on the mesh grid. The black box indicates the kernel size of extension and sampling.

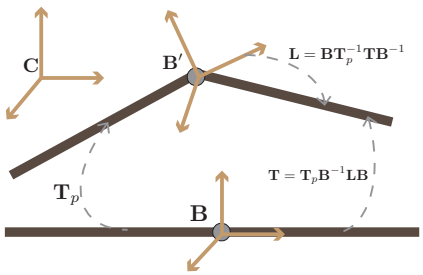


Fig. 5: The relationship between local transformation $\mathbf{L}$ *w.r.t.* the local bone frame $\mathbf{B}$ and global transformation $\mathbf{T}$ *w.r.t.* the camera frame $\mathbf{C}$.

3.4 Refining Mesh Vertices

Post-sampling, the initial mesh estimate is not very accurate (see Fig. 1). But given that our interest is to work with a specific model, *i.e.*, that of the (canonical) hand, it is excessive to add further network structures for more accurate

estimates. Instead, we propose to refine the vertices with a kinematic module. We align the initial mesh estimate with a template mesh model and solve for a rigid transformation via a closed form solution.

More specifically, given the correspondence between estimated vertices $\mathcal{P}_s$ and vertices from the template model $\mathcal{Q}$ for each hand part (palm or finger bone), we estimate a similarity transformation matrix $\mathbf{T}$ by minimizing the Euclidean distance between correspondence points $p_i \in \mathcal{P}_s$ and $q_i \in \mathcal{Q}$ as

$$\mathbf{T}^* = \text{argmin}_{\mathbf{T}} \sum_i \|p_i - \mathbf{T}q_i\|. \tag{3}$$

The refined mesh results from posing the template mesh with the similarity transformation matrices through linear blend skinning (LBS). Note that Eq. 3 is a least squares minimization and that $\mathbf{T}^*$ can be found in closed form [44] *e.g.*, with singular value decomposition (SVD).

By using a closed form solution, the mesh can be refined with a single forward pass through the network. Coordinates of key points can also be obtained from the transformation matrices in a similar way as mesh vertices. And because SVD is differentiable, supervision can also be placed on top of the key-point coordinates. As will be shown in Sec. 5, when given only the supervision of these sparse key-points, our method can accurately recover dense meshes.

3.5 Supervised training loss

We apply MSE to the correspondence estimation $\mathcal{I}_m$ and refined mesh vertices $\mathcal{P}_r$, to optimize network parameters θ, where $\widehat{\mathcal{I}_m^{(i)}}$ and $\widehat{\mathcal{P}_r^{(i)}}$ are the ground-truth correspondence map and mesh vertex coordinates for the ith sample respectively:

$$L(\theta) = \sum_i \|\mathcal{I}_m^{(i)} - \widehat{\mathcal{I}_m^{(i)}}\|^2 + \alpha \|\mathcal{P}_r^{(i)} - \widehat{\mathcal{P}_r^{(i)}}\|^2. \tag{4}$$

3.6 Implementation Details

The hand region is first localized with the segmentation network of [54]. The image input to the hourglass network on the image grid is 64×64; the size of the mesh grid is set as 16×16. To further reduce computation, we adopt pixel shuffling techniques [41] to decrease the spatial resolution by a factor of 2 on both the image grid and mesh grid. While the number of input and output feature channels are increased by a factor of 4, the number of feature channels in hidden layers are unchanged. The kernel size of extension and sampling are both 8×8.

4 Self-supervision on unlabelled real data

Training of the network proposed in Section 3 with direct supervision would require labels in the form of dense correspondences and vertex locations. This

is impossible to annotate for real-world data. Yet training with only synthetic data is also not an option. As shown later in the experiments and also observed in the literature [36, 32, 59], the large domain gap between real and synthesized depth maps gives rise to compromised accuracy. Since our network essentially performs a (differentiable) rendering, the natural question that arises is whether we can incorporate a model-fitting loss into training for self-supervised learning.

The self-supervision term is similar to conventional model-fitting energy functions and is formulated as follows,

$$L(\theta) = \sum_i l^{(i)}_{\text{data}}(\theta) + \lambda_1 l^{(i)}_{\text{prior}}(\theta) + \lambda_2 l^{(i)}_{\text{mv}}(\theta) \tag{5}$$

where θ is the network parameter and $l^{(i)}$ is the loss for the i^{th} sample. For notation simplicity, we omit the superscript in the rest of this section. The term l_{data} measures how much the rendered depth map resembles the input depth map. Priors l_{prior} constrain the estimate to be kinematically feasible. Finally, a multi-view consistency term l_{mv} which can be used in calibrated multi-camera setups to handle self-occlusion. The λ's are associated weighting hyperparameters.

4.1 Data Terms

For l_{data}, we use only an ICP and a lifting energy term:

$$l_{\text{data}}(\theta) = l_{\text{ICP}}(\theta) + \omega l_{\text{lifting}}(\theta). \tag{6}$$

The **ICP term** measures the disparity between points to their projections onto the mesh surface:

$$l_{\text{ICP}}(\theta) = \sum_{i \in \mathcal{I}} \min_{j \in m(\{\mathbf{T}\}|\theta)} d(i, j), \tag{7}$$

where $m(\{\mathbf{T}\}|\theta)$ is the skinned mesh surface, where $\{\mathbf{T}\}$ is a set of per-joint transformation matrices, which are estimated as per Sec 3.4. $l_{\text{ICP}}(\theta)$ approximates the point to surface distance by finding the nearest vertex from the mesh model based on the distance function d. For $d(\cdot, \cdot)$, we use a smooth L_1 loss. Similar to [49], we restrict the points to find only correspondences on the frontal surface of the mesh.

We also leverage the correspondence map and minimize the distance between points and their estimated correspondences on the mesh surface via a **lifting term**:

$$l_{\text{lifting}}(\theta) = \sum_{i \in \mathcal{I}} d(i, f(i|\theta)), \tag{8}$$

where $f(i|\theta)$ estimates the 3D coordinates of the correspondence of i on the mesh surface, given the estimated mesh coordinate of i through the sampling process (see Fig. 4). The lifting term simultaneously optimizes over the correspondence map $\mathcal{I}_m$ on the image grid and the coordinate map $\mathcal{J}_o$ on the mesh grid (see Fig. 2); this introduces more efficient gradient flows to different network stages.

4.2 Kinematic Priors

The kinematic priors are defined as

$$l_{\text{prior}}(\theta) = l_{\text{collision}}(\theta) + \kappa_1 l_{\text{arap}} + \kappa_2 l_{\text{offset}}(\theta). \tag{9}$$

The **collision term** $l_{\text{collision}}(\theta)$ penalizes collisions between any pair of joints:

$$l_{\text{collision}}(\theta) = \sum_{i,j} \max(t - \|p_i - p_j\|, 0), \tag{10}$$

where p_i and p_j are the 3D coordinate of the corresponding joints. We set the threshold $t = 5$mm for all pair of joints.

The **as rigid as possible term** $L_{\text{arap}}(\theta)$ [45] constrains local deformations of estimated mesh surfaces to be rigid:

$$l_{\mathbf{arap}} = \|\mathcal{P}_r - \mathcal{P}_s\|^2, \tag{11}$$

where $\mathcal{P}_s$ are the originally estimated mesh vertices. $\mathcal{P}_r$ are the refined vertices through linear blend skinning and are guaranteed to be rigid for each part.

Section 3.4 described how to estimate the similarity transformation $\mathbf{T}$ with respect to the camera frame for each hand part. $\mathbf{T}$ transforms the bone from a rest pose[1] to the observed pose with respect to the camera frame. From the perspective of forward kinematics, $\mathbf{T}$ can be defined as

$$\mathbf{T} = \mathbf{T}_p \cdot \mathbf{B}^{-1} \cdot \mathbf{L} \cdot \mathbf{B}, \tag{12}$$

where $\mathbf{T}_p$ is the parent transformation matrix, $\mathbf{B}$ is the bone frame in the neutral pose (see Fig. 5) . $\mathbf{L}$ is the local transformation matrix with respect to the bone frame $\mathbf{B}$. Since $\mathbf{B}$ is given in the original mesh model and $\mathbf{T}_p$ is known from previous estimates, $\mathbf{L}$ can be recovered with a closed form solution.

We rewrite $\mathbf{L}$ as $[\mathbf{SR}|t]$, where $\mathbf{S} \in R^{3\times3}$ is a diagonal matrix scaling the matrix, $\mathbf{R} \in R^{3\times3}$ is the rotation matrix, $t \in R^3$ is the translation. Note that except for the wrist, there is no translation on the remaining joints. We thus penalize translations in the finger's local transformation with an **offset term**

$$l_{\text{offset}} = \sum_{i\in\mathcal{F}} \|t_i\|^2, \tag{13}$$

where $\mathcal{F}$ represents all the finger joints.

As the joint angles can be calculated from local transformation $\mathbf{L}$ with a closed-form solution, further constraints such as push constraints can easily be added. We find this to be unnecessary since synthetic data with supervision is also fed to the network to regularize the estimates (see Sec. 4.4).

[1] Defined by placing origin at the joint and aligning the z-axis with its parent bone.

4.3 Multiple view consistency

To handle severe self-occlusions and holes in noisy depth inputs, we add consistency constraints l_{mv} applied to real data captured on a multi-camera rig:

$$l_{\text{mv}}(\theta) = l_{\text{vertex}}(\theta) + \eta_1 l_{\text{ICP}}(\theta) + \eta_2 l_{\text{lifting}}(\theta). \tag{14}$$

By calibrating the extrinsics of the camera, the **vertex term** l_{vertex} minimizes the distance between mesh vertices to their robust average (median in this paper) in the canonical frame. l_{ICP} and l_{lifting} work similarly to the aforementioned single-view cases, except that the estimated mesh model is first mapped to another camera frame and then matched against the corresponding depth map.

4.4 Active data augmentation by estimation

Since the proposed method could recover the hand mesh, we propose a strategy to actively feed synthesized data given the estimated mesh on real data to the network. The supervision from the synthesized data provides more realistic poses and helps the network to better recover from wrong estimates. From our experiments, we find this strategy to be useful to stabilize the self-supervision training and further decrease the model fitting error on unlabelled training data.

5 Experimentation

5.1 Dataset and evaluation protocols

We evaluate on the NYU Hand Pose Dataset [55]. It is currently the only publicly available multi-view depth dataset and features sequences captured by 3 calibrated and synchronized PrimeSense cameras. It consists of 72757×3 frames for training and 8252×3 for testing. NYU is highly challenging as the depth maps are noisy and the sequences cover a wide range of hand poses. Additionally, we synthesize a dataset of 20K depth maps of various hand poses with random holes and noise to evaluate the trained network's ability to generalize to new synthesized samples. We follow [54] to detect hands ($\sim$1ms per frame). In total, our method is highly efficient and achieves 59.2 FPS on an Nvidia 1080Ti GPU.

While our framework is flexible to any hand model, *e.g.*, the MANO model[38], we follow [55] and use the LibHand model from [58] in the following experiments. This provides for an unbiased quantitative analysis since the definition of the palm center differs in different skeleton models. Note that the original hand shape from LibHand is different from either subject in the NYU dataset. Following the protocol of [55] and previous works, we quantitatively evaluate a subset of 14 joints with two standard metrics: mean joint position error (in mm) averaged over all joints and frames, and the percentage of success frames, *i.e.*, frames where all predictions are within a certain threshold [52].

5.2 Training with only synthesized data

We first evaluate how a network trained on synthesized data can generalize to newly synthesized data and real data (see second to sixth row in Table 1). The synthesized data is rendered from a mesh model with various poses and shapes and then corrupted with random depth noise and holes. Data is synthesized in an on-line manner and around 7.2 million samples are fed into the network for training. Our proposed kinematic module successfully reduces the average error over all mesh vertices from 14.75mm to 7.65mm. The network can also generalize to newly synthesized samples and achieves a high accuracy with only 7.1mm mean joint position error. However, the error increases almost three-fold to 23.21mm when testing on real-world depth maps. This shows that even though the network encounters data augmented with random noise, it readily over-fits to the rasterization artifacts and hand shapes of synthesized depth maps.

5.3 Ablation studies

Variations in training data. We investigate how different training data and different supervision impacts the accuracy. First, we train only with the 8252×3 testing samples to check how well self-supervision can fit the mesh model to depth maps. We then trained with all training data, but in a single view setting to check how a multi-view set up impacts performance. Finally, we also look into supervision with sparse key-points to check if the proposed network accurately recover the mesh vertices and the key-points on unseen samples in testing set.

According to Tab. 1, self-supervision based fine tuning on real data significantly reduces the mean joint error from 23.21mm of synthetic data trained network to 16.96mm. Similar improvements can also be found in Fig. 6a with 15% - 20% more successful frames on the error thresholds between 20mm to 40mm after fine tuning. However, single view only is not adequate to address the challenges from noisy depth map and severe self occlusion. To this end, we find leveraging multiple view consistency as additional constraints(see Sec. 4.3) further improve the self-supervision results (see Tab. 1 and Fig. 6a).

Our estimates are highly accurate, with only 8.5mm mean joint position error (see Table 1). Furthermore, 67.8% of frames have a maximum error below 20mm and 85.3% below 30mm respectively (see Fig. 6a). Interestingly, training directly on the test samples gives rise to a higher mean joint error than training on a larger training set excluding the test samples (14.50mm vs 13.09mm, see Table 1). We attribute this to the poor initialization of the network when trained on synthesized data. The learning likely gets trapped in local minima since first-order based optimization is used during back-propagation. However, if the amount of training data increases, mean joint position error decreases. This justifies the benefits of data-driven approaches over conventional model-based trackers which optimizes each frame independently.

As shown in Fig. 1, our method can accurately reconstruct the 3D mesh model given only sparse key-point supervision. When it comes to mean joint position error, the estimation is highly accurate with only 8.5 mean joint position error

(see Table 1). Furthermore, 67.8% of frames have a maximum error below 20mm and 85.3% below 30mm respectively (see Fig. 6a).

Studying the **impact of self-supervision loss terms.** We study the individual contributions of the different self-supervision loss terms by training without the L_{lifting}, $L_{\text{collision}}$, L_{arap}, L_{offset} and active augmentation techniques. The contributions of each of the terms are validated as we observe similar decreases in accuracy when they are omitted (see Table 1 and Fig. 6b). Notable is the fact that without the lifting energy term, the average error increases by 1.41mm from 13.09mm to 14.50mm. The percentage of successful frames drops by 7% from 64% to 57% on the error threshold of 30mm.

5.4 Comparison to state-of-the-art

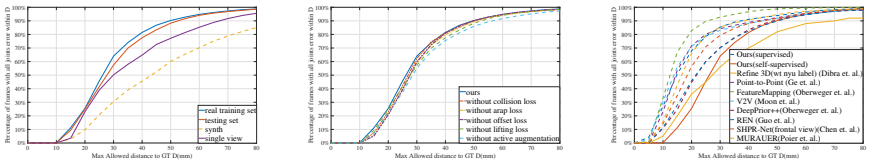
We compare to recent state-of-the-art in Table 2. When trained with keypoint annotations, our method outperforms all other methods except [24] and [36] with respect to mean joint position error. In addition, according to Fig. 6c, our method performs similarly to [14, 32] when the error threshold is larger than 10mm and outperforms all other methods except [36]. We note however that [24] report an ensemble prediction result. This is impractical for real time use; in comparison, our method is highly efficient and runs at 59.2 FPS on an NVidia 1080Ti GPU. Furthermore, we out-perform [24] when compared its single model result. The work of [36] leverages domain adaptation techniques to better utilize synthesized data. This is complementary to our proposed method and beyond our current scope. It is also worth noting that key-point estimation is a byproduct of our proposed method. Our method is not designed to learn key-points; rather, the primary aim of our work is to recover mesh vertices.

We also compare our self-supervision method with [10], which to best of our knowledge is the only other unsupervised method. As is shown in Fig. 6c, our network outperforms [10] by a large margin for the percentage of successful frames at error thresholds higher than 25mm. We achieve a higher accuracy for two reasons. First, our mesh parameterization allows the method to be robust to small estimation offsets while [10] uses joint angles, which tend to propagate errors from parent joints to children joints. Second, there are no gradients in their *depth term*(Eq. 6 in [10]) associated with unexplained points from the depth map which we handle with our proposed data term.

We further compare our self-supervision method with fully supervised deep learning methods. Surprisingly, when trained without any human label, our self-supervision based method achieves competitive results and even out-performs several fully supervised methods[15, 12, 23, 64, 60, 29, 69]. This highly encouraging results suggests that our method can be applied to provide labels for RGB datasets with weak supervision from depth maps.

6 Conclusion and Discussion

We have presented a new network architecture to regress mesh vertices from single depth map with efficient 2D fully convolutional network. At its core is



(a) (b) (c)

Fig. 6: (a) Impact of data used for self-supervision; (b) Impact of different loss terms and active data augmentation on self-supervised learning; (c) Comparison to fully supervised (dashed line) and self-supervised (solid line) state-of-arts.

Method	Mean joint error	Method	Mean joint error
ours (fully supervised)	8.5mm	ours (self-supervised)	13.09mm
variations on training data		**impact of loss terms**	
trained on synt:		without active augmentation	14.52mm
key-points (tested on real)	23.21mm	without L_{lifting}	14.50mm
key-points (tested on sync)	7.10mm	without $L_{\text{collision}}$	13.85mm
mesh vertices (tested on sync)	14.75mm	without L_{arap}	14.06mm
refined mesh vertices (tested on sync)	7.65mm	without L_{offset}	14.12mm
self-supervised learning			
trained on test set	14.50mm		
trained with single view	16.96mm		

Table 1: **Ablation study and self comparison.** We report mean joint error averaged over all joints and frames.

re-parameterization of the mesh model. We demonstrate on-par performance to state-of-arts method in the supervised setting and competitive self-supervision results with multi-camera setup. As future work, we will check how explicit hand shape calibration as proposed in [18] can be incorporated into current framework, as well as extension to RGB inputs.

Acknowledgements. The authors gratefully acknowledge supports from ETH Computer Vision Lab's institutional funding, the Chinese Scholarship Council and the NUS Startup Grant R-252-000-A40-133.

Method	mean joint error	Method	mean joint error
ours (supervised)	8.5mm	ours (self-supervised)	13.1mm
A2J[63]	8.6mm	FeatureMapping[36]	7.4mm
V2V(ensemble)[24]	8.4mm	V2V(single model)[24]	9.2mm
Point-to-Point[14]	9.0mm	SHPR(three views)[8]	9.4mm
MURAUER[32]	9.5mm	DenseReg[61]	10.2mm
Pose-REN[7]	11.8mm	DeepPrior++[27]	12.2mm
REN-4x6x6[15]	13.4mm	3DCNN[12]	14.1mm
DeepHPS(fine-tuned)[23]	14.2mm	Lie-X[64]	14.5mm
CrossingNet[60]	15.5mm	Feedback[29]	15.9mm

Table 2: **Comparison with fully supervised state-of-the-art.** We report mean joint error averaged over all joints and frames. All methods are tested on the NYU[55] test set. We show the comparison for reference, but would like to stress that results are not directly comparable as our method is primarily designed for mesh vertex recovery and not keypoint accuracy.

References

1. Alp Guler, R., Trigeorgis, G., Antonakos, E., Snape, P., Zafeiriou, S., Kokkinos, I.: Densereg: Fully convolutional dense shape regression in-the-wild. In: CVPR (2017)
2. Atzmon, M., Maron, H., Lipman, Y.: Point convolutional neural networks by extension operators. ACM Transactions on Graphics (TOG) (2018)
3. Bogo, F., Kanazawa, A., Lassner, C., Gehler, P., Romero, J., Black, M.J.: Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In: European Conference on Computer Vision. pp. 561–578. Springer (2016)
4. Borg, I., Groenen, P.: Modern Multidimensional Scaling. Springer Series in Statistics, Springer New York (1997)
5. Boukhayma, A., de Bem, R., Torr, P.H.: 3d hand shape and pose from images in the wild. In: CVPR (2019)
6. Cai, Y., Ge, L., Cai, J., Yuan, J.: Weakly-supervised 3d hand pose estimation from monocular rgb images. ECCV, Springer **12** (2018)
7. Chen, X., Wang, G., Guo, H., Zhang, C.: Pose guided structured region ensemble network for cascaded hand pose estimation. arXiv preprint arXiv:1708.03416 (2017)
8. Chen, X., Wang, G., Zhang, C., Kim, T.K., Ji, X.: Shpr-net: Deep semantic hand pose regression from point clouds. IEEE Access **6**, 43425–43439 (2018)
9. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems (2016), https://arxiv.org/abs/1606.09375
10. Dibra, E., Wolf, T., Oztireli, C., Gross, M.: How to refine 3d hand pose estimation from unlabelled depth data? In: 3D Vision (3DV) (2017)
11. Ge, L., Cai, Y., Weng, J., Yuan, J.: Hand pointnet: 3d hand pose estimation using point sets. In: CVPR (2018)
12. Ge, L., Liang, H., Yuan, J., Thalmann, D.: 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In: CVPR. vol. 1, p. 5 (2017)
13. Ge, L., Ren, Z., Li, Y., Xue, Z., Wang, Y., Cai, J., Yuan, J.: 3d hand shape and pose estimation from a single rgb image. In: CVPR (2019)
14. Ge, L., Ren, Z., Yuan, J.: Point-to-point regression pointnet for 3d hand pose estimation. ECCV (2018)
15. Guo, H., Wang, G., Chen, X., Zhang, C., Qiao, F., Yang, H.: Region ensemble network: Improving convolutional network for hand pose estimation. In: Image Processing (ICIP) (2017)
16. Hasson, Y., Varol, G., Tzionas, D., Kalevatykh, I., Black, M.J., Laptev, I., Schmid, C.: Learning joint reconstruction of hands and manipulated objects. In: CVPR (June 2019)
17. Joo, H., Simon, T., Sheikh, Y.: Total capture: A 3d deformation model for tracking faces, hands, and bodies. In: CVPR. pp. 8320–8329 (2018)
18. Joseph Tan, D., Cashman, T., Taylor, J., Fitzgibbon, A., Tarlow, D., Khamis, S., Izadi, S., Shotton, J.: Fits like a glove: Rapid and reliable hand shape personalization. In: CVPR (2016)
19. Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: Computer Vision and Pattern Regognition (CVPR) (2018)
20. Kostrikov, I., Jiang, Z., Panozzo, D., Zorin, D., Joan, B.: Surface networks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018 (2018)

21. Lombardi, S., Saragih, J., Simon, T., Sheikh, Y.: Deep appearance models for face rendering. ACM Transactions on Graphics (TOG) (2018)
22. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia) **34**(6), 248:1–248:16 (Oct 2015)
23. Malik, J., Elhayek, A., Nunnari, F., Varanasi, K., Tamaddon, K., Héloir, A., Stricker, D.: Deephps: End-to-end estimation of 3d hand pose and shape by learning from synthetic depth. In: 2018 International Conference on 3D Vision (3DV) (2018)
24. Moon, G., Chang, J.Y., Lee, K.M.: V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In: CVPR (2018)
25. Mueller, F., Davis, M., Bernard, F., Sotnychenko, O., Verschoor, M., Otaduy, M.A., Casas, D., Theobalt, C.: Real-time pose and shape reconstruction of two interacting hands with a single depth camera. ACM Transactions on Graphics (TOG) **38**(4), 49 (2019)
26. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision (2016)
27. Oberweger, M., Lepetit, V.: Deepprior++: Improving fast and accurate 3d hand pose estimation. In: ICCV workshop (2017)
28. Oberweger, M., Riegler, G., Wohlhart, P., Lepetit, V.: Efficiently creating 3d training data for fine hand pose estimation. In: CVPR. pp. 4957–4965 (2016)
29. Oberweger, M., Wohlhart, P., Lepetit, V.: Training a feedback loop for hand pose estimation. In: ICCV (2015)
30. Omran, M., Lassner, C., Pons-Moll, G., Gehler, P., Schiele, B.: Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In: 2018 International Conference on 3D Vision (3DV). pp. 484–494. IEEE (2018)
31. Pavlakos, G., Zhu, L., Zhou, X., Daniilidis, K.: Learning to estimate 3D human pose and shape from a single color image. In: CVPR (2018)
32. Poier, G., Opitz, M., Schinagl, D., Bischof, H.: Murauer: Mapping unlabeled real data for label austerity. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 1393–1402. IEEE (2019)
33. Poier, G., Schinagl, D., Bischof, H.: Learning pose specific representations by predicting different views. In: CVPR (2018)
34. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593 (2016)
35. Qian, C., Sun, X., Wei, Y., Tang, X., Sun, J.: Realtime and robust hand tracking from depth. In: CVPR (2014)
36. Rad, M., Oberweger, M., Lepetit, V.: Feature mapping for learning fast and accurate 3d pose inference from synthetic images. In: CVPR (2018)
37. Ranjan, A., Bolkart, T., Sanyal, S., Black, M.J.: Generating 3d faces using convolutional mesh autoencoders. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 704–720 (2018)
38. Romero, J., Tzionas, D., Black, M.J.: Embodied hands: Modeling and capturing hands and bodies together. ACM Transactions on Graphics, (Proc. SIGGRAPH Asia) **36**(6) (Nov 2017)
39. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. arXiv preprint arXiv:1905.05172 (2019)

40. Sharp, T., Keskin, C., Robertson, D., Taylor, J., Shotton, J., Kim, D., Rhemann, C., Leichter, I., Vinnikov, A., Wei, Y., et al.: Accurate, robust, and flexible real-time hand tracking. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (2015)
41. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: CVPR (2016)
42. Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., Webb, R.: Learning from simulated and unsupervised images through adversarial training. In: CVPR (2017)
43. Simon, T., Joo, H., Matthews, I.A., Sheikh, Y.: Hand keypoint detection in single images using multiview bootstrapping. In: CVPR (2017)
44. Sorkine, O.: Least-squares rigid motion using svd. Technical notes (2009)
45. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Proceedings of the Fifth Eurographics Symposium on Geometry Processing (2007)
46. Sridhar, S., Mueller, F., Zollhoefer, M., Casas, D., Oulasvirta, A., Theobalt, C.: Real-time joint tracking of a hand manipulating an object from rgb-d input. In: Proceedings of European Conference on Computer Vision (ECCV) (2016)
47. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: SPLATNet: Sparse lattice networks for point cloud processing. In: CVPR. pp. 2530–2539 (2018)
48. Supancic, J.S., Rogez, G., Yang, Y., Shotton, J., Ramanan, D.: Depth-based hand pose estimation: data, methods, and challenges. In: ICCV (2015)
49. Tagliasacchi, A., Schroeder, M., Tkach, A., Bouaziz, S., Botsch, M., Pauly, M.: Robust articulated-icp for real-time hand tracking. Computer Graphics Forum (Symposium on Geometry Processing) **34**(5) (2015)
50. Tan, J., Budvytis, I., Cipolla, R.: Indirect deep structured learning for 3d human body shape and pose prediction. Proceedings of the BMVC, London, UK pp. 4–7 (2017)
51. Tang, D., Taylor, J., Kohli, P., Keskin, C., Kim, T.K., Shotton, J.: Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In: ICCV (2015)
52. Taylor, J., Shotton, J., Sharp, T., Fitzgibbon, A.: The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In: CVPR (2012)
53. Taylor, J., Stebbing, R., Ramakrishna, V., Keskin, C., Shotton, J., Izadi, S., Hertzmann, A., Fitzgibbon, A.: User-specific hand modeling from monocular depth sequences. In: CVPR (2014)
54. Taylor, J., Tankovich, V., Tang, D., Keskin, C., Kim, D., Davidson, P., Kowdle, A., Izadi, S.: Articulated distance fields for ultra-fast tracking of hands interacting. ACM Transactions on Graphics (TOG) (2017)
55. Tompson, J., Stein, M., Lecun, Y., Perlin, K.: Real-time continuous pose recovery of human hands using convolutional networks. ACM Transactions on Graphics (ToG) (2014)
56. Tung, H.Y., Tung, H.W., Yumer, E., Fragkiadaki, K.: Self-supervised learning of motion capture. In: Advances in Neural Information Processing Systems (NIPS) (2017)
57. Varol, G., Ceylan, D., Russell, B., Yang, J., Yumer, E., Laptev, I., Schmid, C.: Bodynet: Volumetric inference of 3d human body shapes. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 20–36 (2018)
58. Šarić, M.: Libhand: A library for hand articulation (2011), http://www.libhand.org/, version 0.9

59. Wan, C., Probst, T., Gool, L.V., Yao, A.: Self-supervised 3d hand pose estimation through training by fitting. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
60. Wan, C., Probst, T., Van Gool, L., Yao, A.: Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation. In: CVPR (2017)
61. Wan, C., Probst, T., Van Gool, L., Yao, A.: Dense 3d regression for hand pose estimation. In: CVPR (2018)
62. Wei, L., Huang, Q., Ceylan, D., Vouga, E., Li, H.: Dense human body correspondences using convolutional networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
63. Xiong, F., Zhang, B., Xiao, Y., Cao, Z., Yu, T., Zhou, J.T., Yuan, J.: A2j: Anchor-to-joint regression network for 3d articulated pose estimation from a single depth image. In: ICCV (2019)
64. Xu, C., Govindarajan, L.N., Zhang, Y., Cheng, L.: Lie-x: Depth image based articulated object pose estimation, tracking, and action recognition on lie groups. International Journal of Computer Vision (2017)
65. Xu, Y., Zhu, S.C., Tung, T.: Denserac: Joint 3d pose and shape estimation by dense render-and-compare. In: ICCV (2019)
66. Yu, R., Saito, S., Li, H., Ceylan, D., Li, H.: Learning dense facial correspondences in unconstrained images. In: CVPR (2018)
67. Yuan, S., Ye, Q., Stenger, B., Jain, S., Kim, T.K.: Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis. In: CVPR (2017)
68. Zhang, X., Li, Q., Zhang, W., Zheng, W.: End-to-end hand mesh recovery from a monocular rgb image. In: ICCV (2019)
69. Zhou, X., Wan, Q., Zhang, W., Xue, X., Wei, Y.: Model-based deep hand pose estimation. arXiv preprint arXiv:1606.06854 (2016)