

 Open access • Book Chapter • DOI:10.1007/978-3-030-58577-8_27

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

— [Source link](#) 

[Chengde Wan](#), [Thomas Probst](#), [Luc Van Gool](#), [Angela Yao](#)





Institutions: [Facebook](#), [ETH Zurich](#), [National University of Singapore](#)

Published on: 23 Aug 2020 - [European Conference on Computer Vision](#)

Topics: [Transformation matrix](#)

Related papers:

- [Embodied hands: modeling and capturing hands and bodies together](#)
- [Learning to Estimate 3D Hand Pose from Single RGB Images](#)
- [What's in a Mesh? A Survey of 3D Mesh Representation Schemes](#)
- [Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images](#)
- [Automatic and interactive mesh to T-spline conversion](#)

Share this paper:    

View more about this paper here: <https://typeset.io/papers/dual-grid-net-hand-mesh-vertex-regression-from-single-depth-408kj7itx6>

Dual Grid Net: hand mesh vertex regression from single depth maps

Chengde Wan¹, Thomas Probst¹, Luc Van Gool^{1,3}, and Angela Yao²

¹ETH Zürich ²National University of Singapore ³KU Leuven

Abstract

We present a method for recovering the dense 3D surface of the hand by regressing the vertex coordinates of a mesh model from a single depth map. To this end, we use a two-stage 2D fully convolutional network architecture. In the first stage, the network estimates a dense correspondence field for every pixel on the depth map or image grid to the mesh grid. In the second stage, we design a differentiable operator to map features learned from the previous stage and regress a 3D coordinate map on the mesh grid. Finally, we sample from the mesh grid to recover the mesh vertices, and fit it an articulated template mesh in closed form.

During inference, the network can predict all the mesh vertices, transformation matrices for every joint and the joint coordinates in a single forward pass. When given supervision on the sparse key-point coordinates, our method achieves state-of-the-art accuracy on NYU dataset for key point localization while recovering mesh vertices and a dense correspondence map. Our framework can also be learned through self-supervision by minimizing a set of data fitting and kinematic prior terms. With multi-camera rig during training to resolve self-occlusion, it can perform competitively with strongly supervised methods without any human annotation.

1. Introduction

We consider the problem of estimating the 3D object shape and pose from single depth images. Specifically, we are interested in estimating the surface mesh vertices of the human hand model from depth maps. Compared to skeleton joints, dense mesh vertices provide both pose and shape information of the hand and enable a much wider scope of virtual and mixed reality applications. For example, one can directly pose the virtual hand in a VR game, or overlay a user’s hand surface with another texture map in mixed reality. Furthermore, the modelling of surface contacts when manipulating virtual objects on screen can be improved with mesh representations.

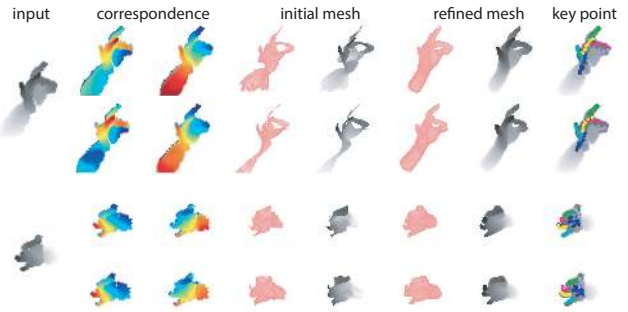


Figure 1. **Qualitative Results.** In each group, upper rows are results supervised with key-point annotation and lower rows are self-supervision result without any human label. We visualize the correspondence map with each mesh coordinate, the rendered shading and depth map of the initial estimated mesh model and refined ones, as well as key-point. More qualitative results will be shown in supplementary material.

The estimation of mesh vertices as opposed to skeleton joints is significantly more challenging in several regards. First, the scale of the problem increases by several magnitudes, *i.e.* to reasonably represent a human hand, one needs thousands of mesh vertices, as opposed to tens of joint positions and angles in a standard skeleton model. 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.

Several recent advances have been made to estimate mesh vertices with deep learning, including the use of voxel net [53], graph convolutions [35, 14], and directly estimating shape parameters and joint angles [6, 60, 22]. These approaches, while having made significant advances for hand pose estimation, have several drawbacks. They tend to be restricted to certain mesh topologies, feature a large number of parameters to learn, and have limited spatial resolutions.

In this work, we propose to solve the problem of mesh vertex regression with a fully convolutional architecture. Our approach is highly efficient and flexible enough to han-

different mesh topologies. Moreover, we can also capture very fine spatial detailing through per-pixel correspondences to a mesh model, thereby allowing for better alignment between the mesh model and depth observations.

We parameterize the mesh vertices with a 2D embedding; the embedding vector associated with each vertex is its “intrinsic” property, *i.e.* only related to its location on the hand mesh surface, regardless of the hand pose, shape and camera view point. In turn, the 3D coordinate of the mesh vertex is considered “extrinsic”. Similar to digital imaging, we discretize the embedding space by placing a 2D grid, namely the *mesh grid* on the mesh embedding. Both “intrinsic” and “extrinsic” properties for each mesh vertex can be approximated in terms of a weighted sum with properties of its neighbour points on the grid.

At the core of our method are two 2D fully convolutional networks, applied to the image and mesh estimates consecutively (see overview in Figure 2). Linking the networks together is a 2D embedding which makes for an efficient way to propagate errors directly from the irregular representation of a mesh to the regular and ordered representation of an image. To refine the estimated mesh, we design a simple kinematic module. Given a template hand mesh model with one-to-one correspondences to the estimated mesh, we solve for a similarity transform through singular value decomposition (SVD). We then re-pose the template mesh based on the transform, resulting in 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 top of the estimated key points.

For training our model, 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.

Our contributions can be summarized as follows,

- We propose a new fully convolutional network architecture for regressing thousands of mesh vertices in an end-to-end manner. While our method works with single depth maps, the network architecture is ready to handle RGB image case without any additional changes.
- 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.

- We bridge the gap between data-driven discriminative methods and optimization-based model-fitting and enjoy benefits from both sides: accuracy that improves with the amount of data encountered, while not needing human-provided 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 the development of ever deeper and more sophisticated neural network architectures [8, 25, 9, 15, 23, 12, 55]. However, such progress has also hinged on the availability of large amounts of annotated data [51, 59, 39]. Obtaining accurate annotations, even for simple 3D joint coordinates, is extremely difficult and time consuming. Annotations generated by manually initializing trackers [51, 26] require carefully designed interfaces for 3D annotation on a 2D screen and there is often little consensus between human annotators [44]. Motion-capture rigs[39] and auxiliary sensors[59] are fully automatic but are limited in the scenes in which they can be deployed. To mitigate the limitations of annotation, semi-supervised approaches [54, 7, 31] and approaches coupling synthesized with real data [38, 30, 34] have also been proposed.

An alternative line of work[49, 33, 45, 36, 47, 18, 42, 50] tackles hand pose estimation 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 tries to bridge the gap between data-driven and model-fitting approaches [52, 11, 14] by using a differentiable renderer and incorporating the model-fitting error as a part of the training loss. Our work resembles these methods, though we have two key differences. First, we re-parameterize the mesh with a 2D embedding, which allows us to use a 2D fully convolutional network architecture. Secondly, we can apply self-supervision on both the image grid and the mesh grid, leading to efficient gradient flow during back-propagation.

Human mesh model recovery from single image. Data-driven methods have greatly advanced the field of 3D reconstruction of both shape and pose of the full human body [48, 56, 4, 28, 46, 29, 52, 19, 53], face [17, 21, 58, 35] and hands [50, 17, 22, 60, 6, 14]. Earlier works were focused on landmark detection[4], segmentation[50] and finding correspondences [48, 56, 17, 58], and performed a model-based optimization to fit the mesh in a subsequent step. However, recent trends have shifted towards end-to-end learning of the mesh with neural networks. For example, [28, 19, 29, 52, 60, 6, 22] directly estimate shape

parameters and joint angles of the mesh. However, such methods are sensitive to perturbations, since small offsets from only one dimension of the estimation easily propagates to many mesh vertices along the kinematic tree. In [46, 21, 35, 14], auto-encoders are used with various decoder structures and outputs, including graph convolution to mesh vertices [35, 14], VoxelNet to 3D occupancy grids[53], and fully connected and transposed convolutions to silhouette [46] and texture and mesh vertices [21]. Unlike any of these works, our approach is based on correspondence estimation. Yet we also differ from other correspondence-based methods [56, 48, 2, 17, 58] in that we estimate mesh vertices with a single forward pass in the framework.

3D Network Architectures. It is highly intuitive to parameterize 3D inputs and or outputs as an occupancy grid or distance field and use for example a 3D voxel net [13, 53, 23]. However, such an architecture is parameter heavy and severely limited in spatial resolution. PointNet [32] 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 a 2D embedding in 3D Euclidean space. As such, several works [10, 20, 35] 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 much simpler and more flexible network architecture which is easier to train and can handle different mesh topologies. Our method most resembles [43, 3] by mapping high dimension data to a 2D grid. However, instead of just working on points from depth map, we propose a dual grid network architecture, enabling the mapping of heterogeneous data from Euclidean space to mesh surfaces and vice versa.

3. Dual Grid Net

In this section, we introduce our Dual Grid Net (DGN) which 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 in a differentiable way.

We assume that we are given a canonical hand mesh model which is generic for all users’ hands. In a given depth map, every pixel on the hand’s surface on the image grid has a correspondence to the mesh surface and estimating this correspondence is equivalent to regressing the pixel’s coordinates on the mesh grid (Sec. 3.1).

Starting from a depth map of the segmented hand as input, the associated mesh vertices can be estimated as fol-

lows. First, we estimate a dense correspondence map to the mesh grid for every point in the input point cloud (see Sec. 3.2). We then map features from the image grid to the mesh grid according to dense correspondence map and recover the 3D coordinates of all the mesh vertices(sec. 3.3). We finally refine these coordinates by skinning a template mesh model with respect to the recovered mesh vertices(sec. 3.4). This process is illustrated in Figure 2.

3.1. Mesh model

We use a triangle mesh model (see Figure 3(a)) with 1721 mesh vertices. Every point on the mesh surface is associated with a mesh coordinate which depends only on its position on the mesh and is therefore invariant to different hand poses, shapes or view point. In addition, other properties of points on the mesh surface such as texture, colour or its 3D coordinates in the camera frame can be approximated with linear interpolation of neighbour points on the mesh surface.

A natural way to parameterize mesh coordinates is through UV mapping [1], as used in [2]. However, the mesh unwrapping in UV mapping introduces unnecessary discontinuities along seams. In this work, we use Multidimensional Scaling (MDS) [5] instead. For any two points on 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 mesh surface. We set the dimension of mesh coordinates to 2, to allow for 2D convolutions on the mesh grid. The MDS embedding used in this work is shown in Figure 3(b), and the corresponding mesh coordinate on mesh surface in Figure 3(c) and (d) respectively.

3.2. Mesh Coordinate Estimation

Similar to [2], we start by estimating the 2D mesh coordinates for all pixels from the hand region. We adopt an hourglass network[24] (see Figure 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 will later be 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 [14, 6, 60, 22] 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 a fully convolutional network architecture. This important difference retains spatial resolution, is more efficient and also translation invariant. It is also much easier for learning, since supervision at the level of mesh coordinates can be directly placed here.

Secondly, the estimated mesh coordinates establishes a dense correspondence map between captured hand surface

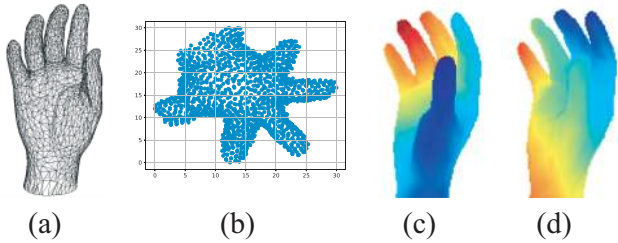
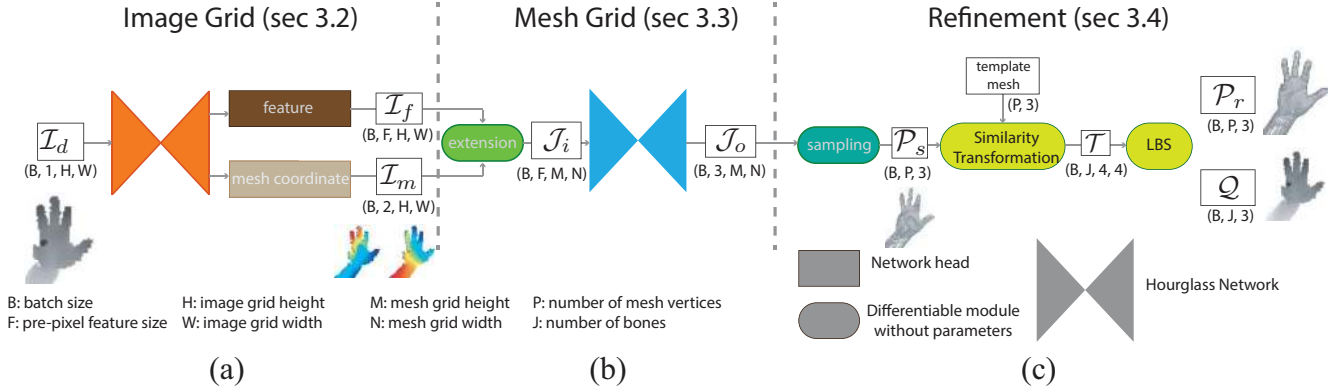


Figure 3. (a) Triangular mesh model used in this work; (b) 2D MDS embedding of the mesh vertices; (c), (d) corresponding mesh coordinates on mesh surface.

to that of mesh. 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 show how to recover all mesh vertices, including occluded ones, from the estimated per-pixel mesh coordinate and features on the image grid. Based on the estimated mesh coordinates, features from a pixel of the hand can be mapped from image grid to mesh grid. Similar to [3], we call this process *extension* (see Figure 4).

More specifically, for any pixel p which belongs 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 described in previous section. f is propagated to mesh grid via soft assignment to the neighbours of m :

$$f = \sum_{n \in \Omega(m)} w_n \cdot f_n. \quad (1)$$

f is propagated to the grid point n with a weighting deter-

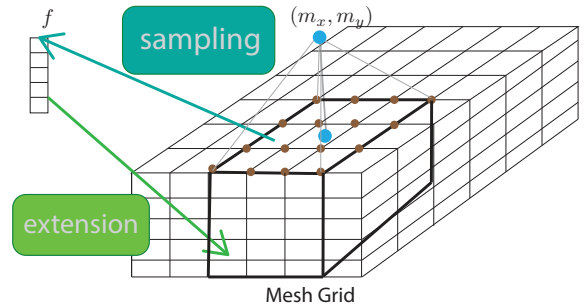


Figure 4. Illustration of extension and sampling process, given the feature to be mapped as $f \in \mathcal{R}^f$ and corresponding coordinate on mesh grid as $(m_x, m_y) \in \mathcal{R}^2$.

mined by the softmax of its distance to m as follows:

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

where $\sigma = 0.5$.

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. To this end, we set the number of output feature channels as 3 and the aggregated feature for each mesh vertices is exactly its estimated 3D coordinates in the camera frame. In turn, this process is named as *sampling* (see Figure 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 inpainting process and we leverage the encoder-decoder structure of hourglass to utilize both global and local context when filling in these values.

3.4. Refining Mesh Vertices

We observe that the quality of the rendered mesh by the estimated mesh is sensitive to even small offsets (see Figure 1). At the same time, as we are focusing on a specific model, it is excessive to add any sophisticated network architectures for more accurate mesh vertices estimation. As an alternative, we propose refining the mesh vertices with a kinematic module without adding learnable parameters.

We refine the estimated mesh vertices by aligning the estimation with a template mesh model and estimating the transformation with 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

$$\mathbf{T} = \arg \min_{\mathbf{T}} \sum_i \|p_i - \mathbf{T}q_i\|. \quad (3)$$

The refined mesh results from posing the template mesh with the similarity transformation matrices through linear blend skinning (LBS). Noticing that \mathbf{T} can also be estimated in closed form with singular value decomposition (SVD). By using the closed form solution, the refined mesh can be obtained with a single forward pass through the network. Readers may refer to [40] for more details on estimating the transformation with a closed form solution.

Coordinates of key points can also be obtained from the transformation matrices in a similar way as mesh vertices. Since SVD is differentiable, supervision can be placed on top of the key-point coordinates. As will be shown in Sec. 5, when only given sparse supervision of key-points, our method can accurately recover the mesh.

3.5. Implementation Details

We first segment the hand region with a hourglass network. The input size of image to the hourglass network on the image grid is 64×64 and we set the size of the mesh grid as 16×16 . To further reduce computation, we adopt pixel shuffling techniques [37] 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 remains unchanged. The kernel size of extension and sampling are both set as 8×8 .

4. Self-supervision on unlabelled real data

Training the network proposed in Section 3 requires supervision in the form of dense correspondences and vertex locations which is impossible to annotate for real world data. While the network can be initialized with synthetic

depth maps, as shown in the experiments, the large domain gap between real and synthesized depth map gives rise to compromised accuracy. On the other hand, since the network also serves as a differentiable renderer, the natural question that arises is whether or not we can include a model-fitting loss term into the training loss for self-supervised learning.

Similar to the conventional model fitting energy, the self-supervision term is formulated as follows,

$$L(\theta) = L_{\text{data}}(\theta) + \lambda_1 L_{\text{prior}}(\theta) + \lambda_2 L_{\text{mv}}(\theta). \quad (4)$$

This data fitting loss is similar to conventional model-fitting energy terms. It is composed of a data term L_{data} , which measures how the rendered depth map resembles the input depth map; kinematic priors L_{prior} which constrain the estimate to be kinematically feasible and a multi-view consistency term L_{mv} which can be used in calibrated multi-camera setups to handle self-occlusion.

4.1. Data Terms

The data term is composed of an ICP term and a lifting energy term:

$$L_{\text{data}}(\theta) = L_{\text{ICP}}(\theta) + \alpha L_{\text{lifting}}(\theta). \quad (5)$$

Similar to [50], we consider only a data-to-model term, *i.e.* only minimizing the distance between every depth point to its correspondence on the mesh surface. Ignoring the model-to-data term makes the loss robust to occlusions which is useful for hand-object or hand-hand interactions.

The **ICP term** measures the disparity between points to its projection on the mesh surface as follows,

$$L_{\text{ICP}}(\theta) = \sum_{i \in \mathcal{I}} \min_{j \in M(\theta)} d(i, j), \quad (6)$$

where the projection on estimated mesh surface $M(\theta)$ is approximated by finding the nearest vertices from mesh model based on the distance function d . We use smooth L1 loss function as $d(\cdot, \cdot)$. Similar to [45], we restrict points only to find correspondences in the frontal surface of the mesh.

In addition, we leverage the correspondence map and minimize the distance between points to their estimated correspondences on the mesh surface via a **lifting term**:

$$L_{\text{lifting}}(\theta) = \sum_{i \in \mathcal{I}} d(i, f(i|\theta)). \quad (7)$$

where $f(i|\theta)$ estimates the 3D coordinates of correspondence of i on the mesh surface, given the estimated mesh coordinate of i through the sampling process (see Figure 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 Figure 2). As such, this helps a more efficient gradient flow to different network stages.

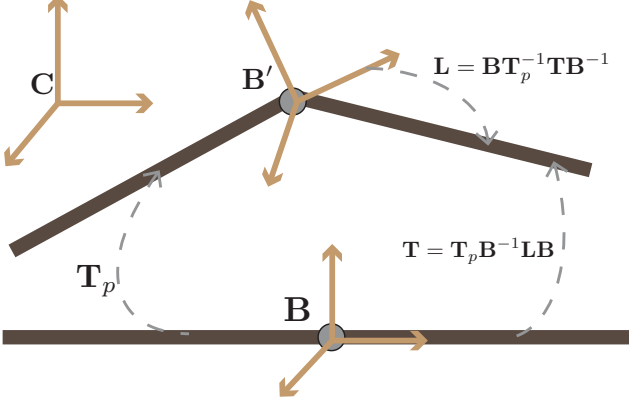


Figure 5. Illustration of the relationship between local transformation \mathbf{L} with respect to the local bone frame \mathbf{B} and global transformation \mathbf{T} with respect to the camera frame \mathbf{C} .

4.2. Kinematic Priors

The kinematic prior terms are defined as

$$L_{\text{prior}}(\theta) = L_{\text{collision}}(\theta) + \kappa_1 L_{\text{arap}} + \kappa_2 L_{\text{offset}}(\theta). \quad (8)$$

The **collision term** $L_{\text{collision}\theta}$ penalizes collisions between any pair of joints as follows:

$$L_{\text{collision}}(\theta) = \sum_{i,j} \max(t - \|p_i - p_j\|, 0), \quad (9)$$

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

The **as rigid as possible term** $L_{\text{arap}}(\theta)$ constrains the local deformation of estimated mesh surfaces to be rigid, similar to [41].

$$L_{\text{arap}} = \|\mathcal{P}_s - \mathcal{P}_r\|^2 \quad (10)$$

where \mathcal{P}_s is the original mesh vertices estimation and \mathcal{P}_r is the refined one through linear blend skinning, which is guaranteed to be rigid for each part.

In section 3.4, we show how to estimate the similarity transformation \mathbf{T} (see Figure 5) with respect to the camera frame for each hand part. In other words, \mathbf{T} transforms the bone from a neutral pose to the current one with respect to the camera frame. From the perspective of forward kinematics, \mathbf{T} is generated as follows,

$$\mathbf{T} = \mathbf{T}_p \cdot \mathbf{B}^{-1} \cdot \mathbf{L} \cdot \mathbf{B}, \quad (11)$$

where \mathbf{T}_p is the parent transformation matrix, \mathbf{B} is the bone frame in the neutral pose with which z-axis is aligned with its parent bone, the origin is placed at the joint. \mathbf{L} is the rotation 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 estimation, \mathbf{L} can be recovered with a closed form solution.

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

$$L_{\text{offset}} = \sum_{i \in \mathcal{F}} \|t_i\|^2, \quad (12)$$

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

We don't add further push constraints over the joint angles since synthesized data with supervision is also fed to the network to regularize the estimation. Given that joint angles can be calculated from local transformation \mathbf{L} with a closed form solution, joint angle constraints can be easily added if necessary.

4.3. Multiple view consistency

To handle severe self-occlusions and missing inputs due to holes in noisy depth inputs, we further add multi-view consistency constraints for real data captured on a multi-camera rig:

$$L_{\text{mv}}(\theta) = L_{\text{vertex}}(\theta) + \eta_1 L_{\text{ICP}}(\theta) + \eta_2 L_{\text{lifting}}(\theta). \quad (13)$$

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. The **ICP term** L_{ICP} and **lifting term** L_{lifting} works similarly to the aforementioned single view cases, with the only difference that estimated mesh model is mapped to another camera frame and 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 estimation. According to 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 our method on the NYU Hand Pose Dataset[51]. It currently the only publicly available multi-view depth dataset and features sequences captured by 3 calibrated and synchronized PrimeSense depth 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. In addition, we synthesize a dataset of 20K depth maps of various hand poses with random holes and depth noise to evaluate the trained network’s ability to generalize to new synthesized samples. Our method is highly efficient and achieves 63.1 FPS on an Nvidia 1080Ti GPU.

Following the protocol of [51] 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 with all predictions are within a certain threshold [48]. Readers may refer to supplementary materials and video for qualitative results.

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. 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 synthesized samples are fed into the network for training. Table 3 (synt(test on synt), synt(mesh vertices)(test on synt), synt(refined mesh vertices)(test on synt)) shows that the 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 accuracy deteriorates dramatically when testing on real-world depth maps. The mean average joint error increases almost three-fold to 23.21mm. This shows that even though it encounters data augmented with random noise, the network 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.

Interestingly, training directly on the test samples gives rise to a higher mean joint position error than when training on a larger training set that excludes the test samples (14.50mm vs 13.09mm, see Table 3). We attribute this to the poor initialization of the network when trained on

synthesized data and the possibility of getting 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 Figure 1 (see more qualitative examples in the supplementary materials), 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 3). Furthermore, 67.8% of frames have a maximum error below 20mm and 85.3% below 30mm respectively (see Table 6).

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. According to Table 3 and Figure 6, without the lifting energy techniques, 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. This validates the benefits of the lifting energy. The contributions of the other terms can also be validated as we observe similar decreases in accuracy when they are omitted.

5.4. Comparison to state-of-the-art

We compare our results to the most recently proposed state-of-arts [34, 23, 15, 9, 30, 55, 8, 25, 16, 13, 22, 57, 54, 27, 61]. As shown in Table 2, when trained with key-point annotations, our method outperforms all other state-of-arts except [23] and [34] with respect to mean joint position error. In addition, according to Figure 7, our method performs similarly to [15, 30] when the error threshold is larger than 10mm and outperforms all other methods except [34]. We note however that [23] report an ensemble prediction result. This is impractical for real time use; in comparison, our method is highly efficient and runs at 63.1 FPS on an NVidia 1080Ti GPU. Furthermore, our method outperforms [23] when compared its single model result. The work of [34] leverages domain adaption techniques to better utilize synthesized data. This is complimentary to our proposed method and we look forward to incorporating this in our future work. It is also worth noting that key-point estimation is a byproduct of our proposed method, which has the primary aim of recovering the mesh vertices.

We also compare our self-supervision method with [11], which to best of our knowledge is the only other unsupervised method. As is shown in Figure 7, our network outperforms [11] 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

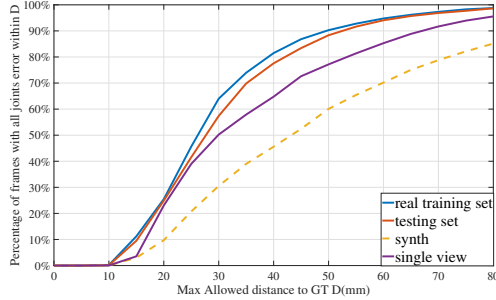


Figure 6. Impact of using different dataset for self-supervision.

parameterization allows the method to be robust to small estimation offsets while [11] 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 [11]) 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 [16, 13, 22, 57, 54, 27, 61]. This highly encouraging results suggests that our method can be applied to provide labels for RGB datasets with weak supervision from depth maps.

Method	Mean joint error
ours (fully supervised)	8.5mm
ours (self-supervised)	13.09mm
synt(test on synt)	7.10mm
synt(mesh vertices) (test on synt)	14.75mm
synt(refined mesh vertices) (test on synt)	7.65mm
synt(test on real)	23.21mm
train on test	14.50mm
single view	16.96mm
without active augmentation	14.52mm
without $L_{lifting}$	14.50mm
without $L_{collision}$	13.85mm
without L_{arap}	14.06mm
without L_{offset}	14.12mm

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

6. Conclusion

We propose a new network architecture to regress thousands of mesh vertices from single depth map with efficient fully convolutional network on 2D grids. We show that when initialized with synthesized data, the network could accurately recover the hand mesh vertices with only sparse key point supervision. When given only unlabeled real world dataset, the proposed network can be fine tuned

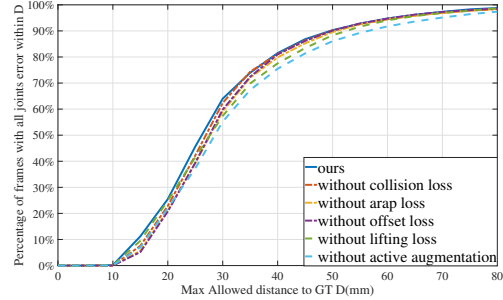


Figure 7. Impact of different loss terms and active data augmentation on self-supervised learning.

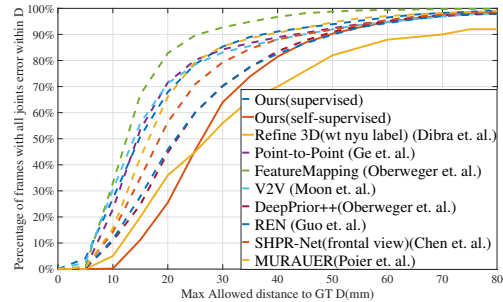


Figure 8. Comparison to fully supervised (dashed line) and self-supervised (solid line) state-of-arts.

Method	mean joint error
ours (supervised)	8.5mm
ours (self-supervised)	13.1mm
FeatureMapping[34]	7.4mm
V2V(ensemble)[23]	8.4mm
V2V(single model)[23]	9.2mm
Point-to-Point[15]	9.0mm
SHPR(three views)[9]	9.4mm
MURAUER[30]	9.5mm
DenseReg[55]	10.2mm
Pose-REN[8]	11.8mm
DeepPrior++[25]	12.2mm
REN-4x6x6[16]	13.4mm
3DCNN[13]	14.1mm
DeepHPS(fine-tuned)[22]	14.2mm
Lie-X[57]	14.5mm
CrossingNet[54]	15.5mm
Feedback[27]	15.9mm
DeepModel[61]	17.0mm

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[51] test set.

in a self-supervision manner and provide comparable accuracy to state-of-arts with multi-camera rig during training. Finally, although this paper focuses on depth map input the human hand, since we use 2D FCN, the proposed method

can be readily applied to RGB images without any changes to the architecture, when RGB annotation is available.

References

- [1] https://en.wikipedia.org/wiki/UV_mapping.
- [2] R. Alp Guler, G. Trigeorgis, E. Antonakos, P. Snape, S. Zafeiriou, and I. Kokkinos. Densereg: Fully convolutional dense shape regression in-the-wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [3] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)*, 2018.
- [4] 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 *European Conference on Computer Vision*, pages 561–578. Springer, 2016.
- [5] I. Borg and P. Groenen. *Modern Multidimensional Scaling*. Springer Series in Statistics. Springer New York, 1997.
- [6] A. Boukhayma, R. de Bem, and P. H. Torr. 3d hand shape and pose from images in the wild. In *CVPR*, 2019.
- [7] Y. Cai, L. Ge, J. Cai, and J. Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. *ECCV, Springer*, 12, 2018.
- [8] X. Chen, G. Wang, H. Guo, and C. Zhang. Pose guided structured region ensemble network for cascaded hand pose estimation. *arXiv preprint arXiv:1708.03416*, 2017.
- [9] X. Chen, G. Wang, C. Zhang, T.-K. Kim, and X. Ji. Shprnet: Deep semantic hand pose regression from point clouds. *IEEE Access*, 2018.
- [10] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, 2016.
- [11] E. Dibra, T. Wolf, C. Oztireli, and M. Gross. How to refine 3d hand pose estimation from unlabelled depth data? In *3D Vision (3DV)*, 2017.
- [12] L. Ge, Y. Cai, J. Weng, and J. Yuan. Hand pointnet: 3d hand pose estimation using point sets. In *CVPR*, 2018.
- [13] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 5, 2017.
- [14] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan. 3d hand shape and pose estimation from a single rgb image. In *CVPR*, 2019.
- [15] L. Ge, Z. Ren, and J. Yuan. Point-to-point regression pointnet for 3d hand pose estimation. *ECCV*, 2018.
- [16] H. Guo, G. Wang, X. Chen, C. Zhang, F. Qiao, and H. Yang. Region ensemble network: Improving convolutional network for hand pose estimation. In *Image Processing (ICIP)*, 2017.
- [17] H. Joo, T. Simon, and Y. Sheikh. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8320–8329, 2018.
- [18] D. Joseph Tan, T. Cashman, J. Taylor, A. Fitzgibbon, D. Tarrow, S. Khamis, S. Izadi, and J. Shotton. Fits like a glove: Rapid and reliable hand shape personalization. In *CvPR*, 2016.
- [19] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [20] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and B. Joan. Surface networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 2018.
- [21] S. Lombardi, J. Saragih, T. Simon, and Y. Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics (TOG)*, 2018.
- [22] J. Malik, A. Elhayek, F. Nunnari, K. Varanasi, K. Tamaddon, A. Héloir, and D. Stricker. Deephps: End-to-end estimation of 3d hand pose and shape by learning from synthetic depth. 2018.
- [23] G. Moon, J. Y. Chang, and K. M. Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *CVPR*, 2018.
- [24] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, 2016.
- [25] M. Oberweger and V. Lepetit. Deepprior++: Improving fast and accurate 3d hand pose estimation. In *ICCV workshop*, 2017.
- [26] M. Oberweger, G. Riegler, P. Wohlhart, and V. Lepetit. Efficiently creating 3d training data for fine hand pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4957–4965, 2016.
- [27] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a feedback loop for hand pose estimation. In *ICCV*, 2015.
- [28] M. Omran, C. Lassner, G. Pons-Moll, P. Gehler, and B. Schiele. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *2018 International Conference on 3D Vision (3DV)*, pages 484–494. IEEE, 2018.
- [29] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis. Learning to estimate 3D human pose and shape from a single color image. In *CVPR*, 2018.
- [30] G. Poier, M. Opitz, D. Schinagl, and H. Bischof. Muraue: Mapping unlabeled real data for label austerity. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1393–1402. IEEE, 2019.
- [31] G. Poier, D. Schinagl, and H. Bischof. Learning pose specific representations by predicting different views. In *CVPR*, 2018.
- [32] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.
- [33] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *CVPR*, 2014.

- [34] M. Rad, M. Oberweger, and V. Lepetit. Feature mapping for learning fast and accurate 3d pose inference from synthetic images. In *CVPR*, 2018.
- [35] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018.
- [36] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, 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.
- [37] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [38] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017.
- [39] T. Simon, H. Joo, I. A. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.
- [40] O. Sorkine. Least-squares rigid motion using svd. *Technical notes*, 2009.
- [41] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, 2007.
- [42] S. Sridhar, F. Mueller, M. Zollhoefer, D. Casas, A. Oulasvirta, and C. Theobalt. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.
- [43] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz. SPLATNet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.
- [44] J. S. Supancic, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan. Depth-based hand pose estimation: data, methods, and challenges. In *ICCV*, 2015.
- [45] A. Tagliasacchi, M. Schroeder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly. Robust articulated-icp for real-time hand tracking. *Computer Graphics Forum (Symposium on Geometry Processing)*, 34(5), 2015.
- [46] J. Tan, I. Budvytis, and R. Cipolla. Indirect deep structured learning for 3d human body shape and pose prediction. *Proceedings of the BMVC, London, UK*, pages 4–7, 2017.
- [47] D. Tang, J. Taylor, P. Kohli, C. Keskin, T.-K. Kim, and J. Shotton. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *ICCV*, 2015.
- [48] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *CVPR*, 2012.
- [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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [50] J. Taylor, V. Tankovich, D. Tang, C. Keskin, D. Kim, P. Davidson, A. Kowdle, and S. Izadi. Articulated distance fields for ultra-fast tracking of hands interacting. *ACM Transactions on Graphics (TOG)*, 2017.
- [51] 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)*.
- [52] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki. Self-supervised learning of motion capture. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.
- [53] G. Varol, D. Ceylan, B. Russell, J. Yang, E. Yumer, I. Laptev, and C. Schmid. Bodynet: Volumetric inference of 3d human body shapes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36, 2018.
- [54] C. Wan, T. Probst, L. Van Gool, and A. Yao. Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation. In *CVPR*, 2017.
- [55] C. Wan, T. Probst, L. Van Gool, and A. Yao. Dense 3d regression for hand pose estimation. In *CVPR*, 2018.
- [56] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [57] C. Xu, L. N. Govindarajan, Y. Zhang, and L. Cheng. Lie-x: Depth image based articulated object pose estimation, tracking, and action recognition on lie groups. *International Journal of Computer Vision*, 2017.
- [58] R. Yu, S. Saito, H. Li, D. Ceylan, and H. Li. Learning dense facial correspondences in unconstrained images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [59] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim. Big-hand2. 2m benchmark: Hand pose dataset and state of the art analysis. In *CVPR*, 2017.
- [60] X. Zhang, Q. Li, W. Zhang, and W. Zheng. End-to-end hand mesh recovery from a monocular rgb image. *arXiv preprint arXiv:1902.09305*, 2019.
- [61] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei. Model-based deep hand pose estimation. *arXiv preprint arXiv:1606.06854*, 2016.

Supplemental Materials

7. Qualitative results

We show more qualitative results on the testing set of NYU dataset in Fig. 10, 11 and 12. Left column shows results trained by the sparse key point supervision. Right column shows results trained by the proposed self-supervision method. Readers may also refer to the attached video to check qualitative results on more frames.

8. Self-supervision training error

We investigate how well the proposed self-supervision method can fit to the training set itself, *i.e.*, the training error, as “self-supervised(test on training set)” in Tab. 3 and Fig. 9. Since our self-supervision method can be potentially applied for automatic annotation of depth frames and accompanied RGBs, its training error indicates how accurate can the annotation be.

We compare the training error against its corresponding testing error, *i.e.*, on the unseen testing set with the same network (“self-supervised”), as well as the testing error trained only with synthesized dataset (“synthesize”), and training error on the testing set (“self-supervised(train on testing set)”), which is roughly 9 times smaller than the training set. As expected, compared to accuracy on the testing set, the mean joint error on training set decreases by 1.2mm from 13.1mm to 11.9mm according to Tab. 3, and around 10% more successful frames on the error threshold between 20 to 40mm according to Fig. 9. When comparing with the recent proposed state-of-arts with complicated network architectures and trained with full supervision[57, 13, 16, 25, 8, 55], our self-supervision method provides with competitive or even higher accuracy. This validates our self-supervision method can provide with highly accurate annotation.

In addition, as already discussed in the paper, the accuracy of self-supervision method is also impacted by the size of the dataset, even when no human label is provided. This infers that accuracy can be further improved when collecting a larger scale dataset.

Method	Mean joint error
self-supervised	13.1mm
synthesize	23.2mm
self-supervised(test on training set)	11.9mm
self-supervised(train on testing set)	14.5mm
Lie-X[57]	14.5mm
3DCNN[13]	14.1mm
REN-4x6x6[16]	13.4mm
DeepPrior++[25]	12.2mm
Pose-REN[8]	11.8mm
DenseReg[55]	10.2mm

Table 3. **Ablation study and self comparison.** We report mean joint error averaged over all joints and frames.

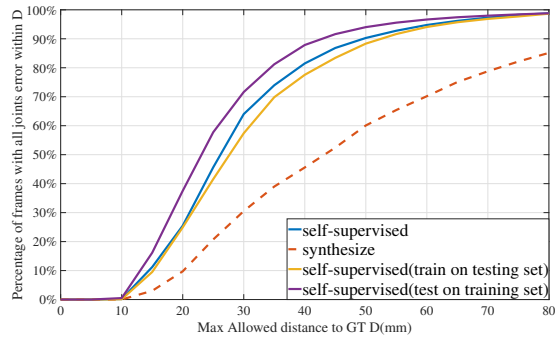


Figure 9. Comparison of using different dataset for training and testing for proposed self-supervision method.

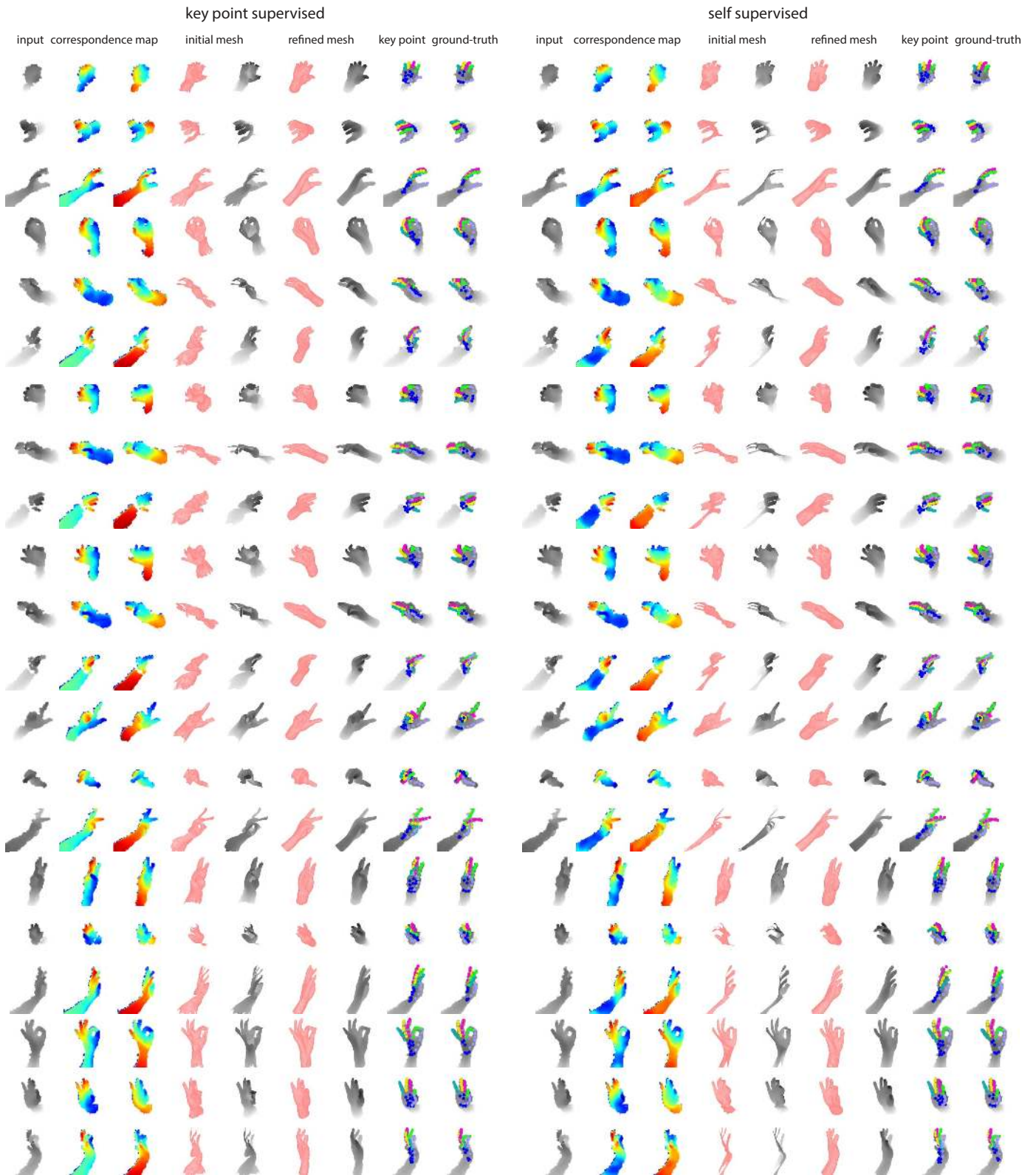


Figure 10. Qualitative results on NYU dataset. We visualize the correspondence map with each mesh coordinate, the rendered shading and depth map of the initial estimated mesh model and refined ones, as well as estimated and ground truth key-point.

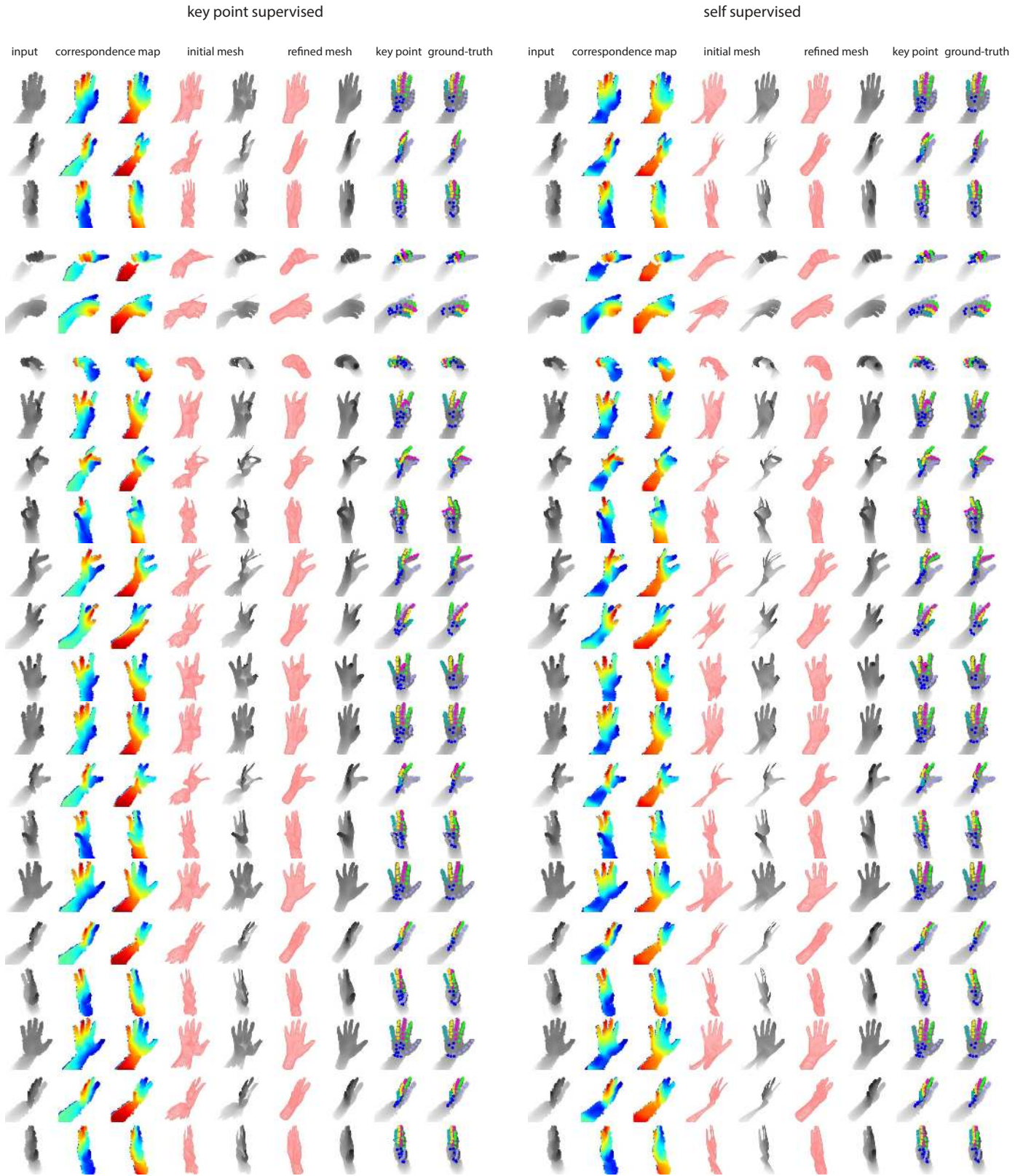


Figure 11. Qualitative results on NYU dataset. We visualize the correspondence map with each mesh coordinate, the rendered shading and depth map of the initial estimated mesh model and refined ones, as well as estimated and ground truth key-point.

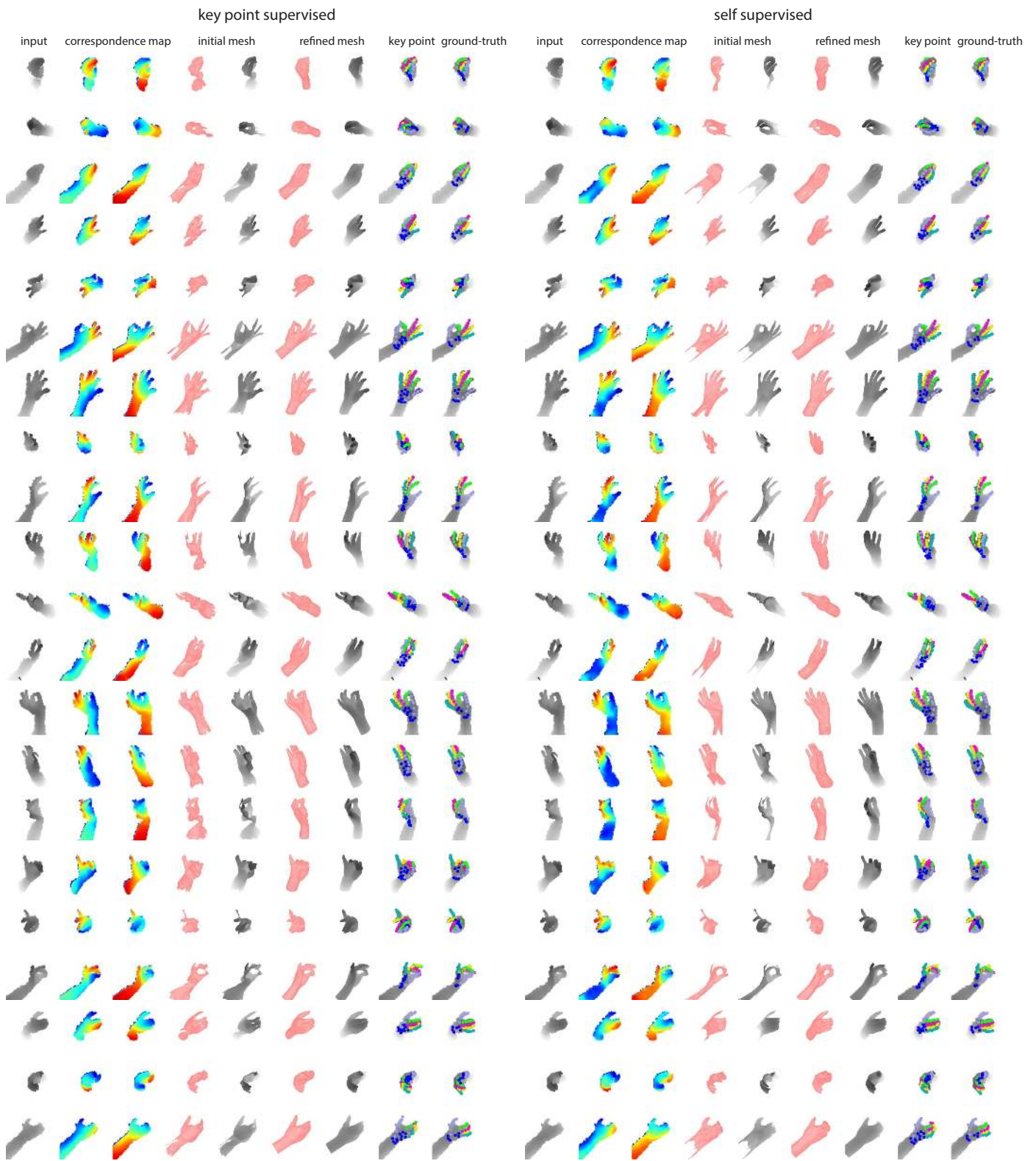


Figure 12. Qualitative results on NYU dataset. We visualize the correspondence map with each mesh coordinate, the rendered shading and depth map of the initial estimated mesh model and refined ones, as well as estimated and ground truth key-point.