HandTailor: Towards High-Precision Monocular 3D Hand Recovery

Jun Lv[12]
lyujune_sjtu@sjtu.edu.cn
Wenqiang Xu[1]
vinjohn@sjtu.edu.cn
Lixin Yang[1]
siriusyang@sjtu.edu.cn
Sucheng Qian[1]
qiansucheng@sjtu.edu.cn
Chongzhao Mao[2]
chongzhao.mao@flexiv.com
Cewu Lu[1]
lucewu@sjtu.edu.cn

[1] Department of Computer Science
Shanghai Jiao Tong University
Shanghai CHINA

[2] Flexiv Ltd.
Shanghai CHINA

Abstract

3D hand pose estimation and shape recovery are challenging tasks in computer vision. We introduce a novel framework HandTailor, which combines a learning-based *hand module* and an optimization-based *tailor module* to achieve high-precision hand mesh recovery from a monocular RGB image. The proposed *hand module* adapts both perspective projection and weak perspective projection in a single network towards accuracy oriented and in-the-wild scenarios. The proposed *tailor module* then utilizes the coarsely reconstructed mesh model provided by the *hand module* as initialization to obtain better results. The *tailor module* is time-efficient, costs only $\sim 8ms$ per frame on a modern CPU. We demonstrate that HandTailor can get state-of-the-art performance on several public benchmarks, with impressive qualitative results. Code and video are available on our project webpage https://sites.google.com/view/handtailor.

1 Introduction

The hand is one of the most important elements of humans when interacting with the environment. Single image 3D hand reconstruction is a task that seeks to estimate the hand model from a monocular RGB image, which could be beneficial for various applications like human behavior understanding, VR/AR, and human-robot interaction.

Recovery of the 3D hand model from a single image has been studied for decades, but challenges remain. First, since the 3D hand reconstruction is a process of 2D-3D mapping, the learning method should somehow handle the camera projection. Some of the previous works [13, 39] choose to ignore this issue by only predicting root-relative hand mesh, which

limits the application of their algorithms. Others rely on perspective projection [15, 35] or weak perspective projection [1, 2, 37]. Perspective projection is more accurate but requires intrinsic camera parameters, making it impossible to apply without camera information. While weak perspective projection can be applied to in-the-wild cases but the approximation is conditioned. For previous works, once the camera projection model is selected, the adaptability of the method is also determined. Second, predicting 3D mesh usually cannot guarantee back-projection consistency, which means, a visual appealing predicted 3D hand may have numerous evident errors, such as a few degrees of finger deviation, when it is re-projected onto the original image (See Fig. 2).

To address these two issues, we propose a novel framework named **HandTailor**, which consists of a CNN-based hand mesh generation module (*hand module*) and an optimization-based tailoring module (*tailor module*). The *hand module* is compatible with both perspective projection and weak perspective projection without any modification of the structure and model parameters. It can be used to project the 3D hand more accurately when the camera parameters are available and also can be used to predict the in-the-wild image by simply changing the computational scheme. The *tailor module* can refine the rough hand mesh predicted by *hand module* to higher precision and fix the 2D-3D miss-alignment based on more reliable intermediate results. With the initialization provided by the *hand module* and the differentiability of the *tailor module*, the optimization adds only $\sim 8ms$ overhead.

Experiments show that HandTailor can achieve comparable results with several state-of-the-art methods, and can accomplish both perspective projection and weak perspective projection. The *tailor module* can improve the performance quantitatively and qualitatively. On a stricter AUC_{5-20} metric, HandTailor gets 0.658 with an improvement close to 0.1 by the *tailor module* on RHD [41]. To prove the applicability and generality of the *tailor module* upon other intermediate representation-based approaches [2, 35, 39], we run several plug-and-play tests, which also show performance improvements by a large margin.

Our contributions can be summarized as follows: First, we propose a novel framework HandTailor for single image 3D hand recovery task, which combines a learning-based *hand module* and an optimization-based *tailor module*. The proposed HandTailor achieves state-of-the-art results among many benchmarks. Second, this method adapts both weak perspective projection and perspective projection without modification of the structure itself. Such architecture can be applied to both in-the-wild and accuracy-oriented occasions. Third, the *tailor module* refines the regressed hand mesh by optimizing the energy function w.r.t the intermediate outputs from the *hand module*. It can also be used as an off-the-shelf plugin for other intermediate representation-based methods. It shows significant improvements both qualitatively and quantitatively.

2 Related Works

In this section, we discuss the existing 3D hand pose estimation and shape recovery methods. There are lots of approaches based on depth maps or point clouds data [9, 10, 19, 21, 32, 36], but in this paper, we mainly focus on single RGB-based approaches.

3D Hand Pose Estimation. Zimmermann and Brox [40] propose a neural network to estimate 3D hand pose from a single RGB image, which lays the foundation for subsequent research. Iqbal et al. [15] utilize a 2.5D pose representation for 3D pose estimation, provide another solution for 2D-3D mapping. Cai et al. [4] propose a weakly supervised method by generating depth maps from predicted 3D pose to gain 3D supervision, which gets rid of 3D

annotations. Mueller et al. [22] use CycleGAN [40] to bridge the gap between synthetic and real data to enhance training. Some other works [13, 28, 33, 34, 38] formulate 3D hand pose estimation as a cross-modal problem, trying to learn a unified latent space.

3D Hand Mesh Recovery. 3D mesh is a richer representation of human hand than 3D skeleton. To recover 3D hand mesh from monocular RGB images, the most common way is to predict the parameters of a predefined parametric hand model like MANO [26]. Boukhayma et al. [2] directly regress the MANO parameters via a neural network and utilize a weak perspective projection to enable the in-the-wild scenes. Baek et al. [1] and Zhang et al. [37] utilize a differential renderer [16] to gain more supervision from hand segmentation masks. These methods generally predict the PCA components of MANO parameters, causing inevitable information loss. To address this issue, Zhou et al. [39] propose IKNet to directly estimate the rotations of all hand joints from 3D hand skeleton. Yang et al. [35] reconstruct hand mesh with multi-stage bisected hourglass networks. Chen et al. [6] achieve camera-space hand mesh recovery via semantic aggregation and adaptive registration. Different from the aforementioned model-based method, there are also some approaches [11, 17, 18] that generate hand mesh through GCN [8], providing new thoughts for this task. [6, 43] try to accomplish this task with self-supervise learning. Different from the aforementioned method, we propose a novel framework that combines a learning-based module and an optimization-based module to achieve better performance, and also adapts both weak perspective projection and weak perspective projection for high precision and in-the-wild scenarios.

Optimization-based 3D Hand Mesh Recovery. Apart from the learning-based methods, there are also some other attempts on reconstructing hand mesh in an optimization-based manner. Previous works choose to fit the predefined hand model [24, 31] to depth maps [24, 27, 30, 31]. For monocular RGB reconstruction, Panteleris et al. [25] propose to fit a parametric hand mesh onto 2D keypoints extracted from RGB image via a neural network [5]. Mueller et al. [22] introduce more constraints for better optimization, like 3D joints locations. Kulon et al. [18] utilize iterative model fitting to generate 3D annotations from 2D skeleton to achieve weakly supervise learning, treating the recovery result of the optimization-based method as the upper bound of the learning-based method. Though these optimization-based methods share a similar ideology to our *tailor module*, our approach exploits the multi-stage design of the *hand module* to make use of information from different stages and accelerates the optimization process. The reduced overhead makes the optimization possible while in inference, which is crucial for practical usage.

3 Method

3.1 Preliminary

MANO Hand Model. MANO [26] is a kind of parametric hand model, which factors a full hand mesh to the pose parameters $\theta \in \mathbb{R}^{16\times 3}$ and the shape parameters $\beta \in \mathbb{R}^{10}$. The hand mesh $\mathcal{M}(\theta,\beta) \in \mathbb{R}^{V\times 3}$ can be obtained via a linear blend skinning function $\mathcal{W}$,

$$\mathcal{M}(\theta,\beta) = \mathcal{W}(\mathcal{T}(\beta,\theta),\mathcal{J}(\beta),\theta,\omega) \tag{1}$$

$\mathcal{T}$ is the rigged template hand mesh with 16 joints $\mathcal{J}$. ω denotes the blend weights, and $V = 778$ is the vertex number of hand mesh. For more details please refer to [26].

Camera Models. Perspective projection describes the imaging behavior of cameras and human eyes. To transform 3D points in camera coordinate to 2D pixels in image plane, we

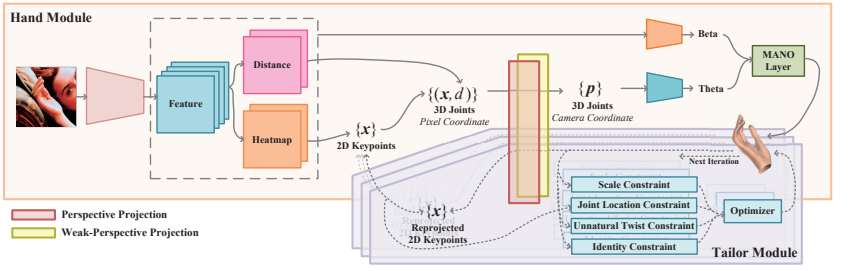


Figure 1: HandTailor consists of two components, *hand module* and *tailor module*. *Hand module* predicts the parameters of MANO through the pose and shape branches to generate hand mesh. It takes a multi-stage design and produces 2D keypoints and 3D joints as intermediate results. The *tailor module* optimizes the predicted hand mesh according to several constraints. HandTailor adapts both perspective projection and weak perspective projection.

need camera intrinsic matrix $\mathcal{K} \in \mathbb{R}^{3\times3}$,

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \pi(\mathcal{K}; \begin{bmatrix} x \\ y \\ z \end{bmatrix}) = \frac{1}{z} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2}$$

$\pi(\cdot)$ is the projection function, (f_x, f_y) are focal lengths, and (c_x, c_y) are camera centers.

Weak perspective projection takes a simplification on the intrinsic matrix, formulated as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \Pi(\mathcal{K}'; \begin{bmatrix} x \\ y \\ z \end{bmatrix}) = \frac{1}{z} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3}$$

$\Pi(\cdot)$ is the weak perspective projection function, $s \in \mathbb{R}$ is the scaling factor. To align the re-projected model to the image, we also need camera extrinsic parameters for rotation $\mathcal{R} \in \mathbb{R}^{3\times3}$ and translation $t \in \mathbb{R}^3$. In practice, we set $\mathcal{R}$ as identity matrix and predict t solely.

3.2 Overview

The proposed HandTailor consists of two components, a learning-based *hand module* (Sec. 3.3) and an optimization-based *tailor module* (Sec. 3.4), which aims to reconstruct the 3D hand mesh from RGB images. The overall pipeline is shown in Fig. 1.

3.3 Hand Module

Intermediate Representation Generation. Given RGB image $\mathcal{I} \in \mathbb{R}^{H\times W\times3}$ we utilize a stacked hourglass network [23] to generate feature space $\mathcal{F} \in \mathbb{R}^{N\times H\times W}$.Then the 2D keypoint heatmaps $\mathcal{H} \in \mathbb{R}^{k\times H\times W}$, and distance maps $\mathcal{D} \in \mathbb{R}^{k\times H\times W}$ are predicted from $\mathcal{F}$. The sum of $\mathcal{H}$ is normalized to 1 at each channel, $\mathcal{D}$ is the root-relative and scale-normalized distance for each joint. We scale the length of the reference bone to 1.

Pose-Branch. This branch is to predict θ. We first retrieve 2D keypoints $\mathcal{X} = \{\boldsymbol{x}_i | \boldsymbol{x}_i = (u_i, v_i) \in \mathbb{R}^2\}_{i=1}^k$ from heatmap $\mathcal{H}$. For j^{th} joint, the 2D keypoint $\boldsymbol{x}_j \in \mathcal{X}$ and the root-relative scale-normalized depth d_j of j^{th} joint in the image plane can be achieved by

$$(\boldsymbol{x_j}, d_j)^\top = (\sum_{\boldsymbol{x}\in\mathcal{H}} H^{(j)}(\boldsymbol{x}) \cdot \boldsymbol{x}^\top, \sum_{\boldsymbol{x}\in\mathcal{H}} H^{(j)}(\boldsymbol{x}) \cdot D^{(j)}(\boldsymbol{x})) \tag{4}$$

$(\boldsymbol{x}_j, d_j) = [u_j,\ v_j,\ d_j]$ is the scaled pixel coordinate of j^{th} joint. The root-relative and scale-normalized depth d_j are friendly for neural network training. To obtain the 3D keypoints $\mathcal{P} = \{\boldsymbol{p}_i | \boldsymbol{p}_i = (x_i, y_i, z_i) \in \mathbb{R}^3\}_{i=1}^{k}$, we project the $[u_j,\ v_j,\ d_j]$ into camera coordinate as $\boldsymbol{p}_j = [x_j, y_j, z_j] \in \mathcal{P}$. To do so, first we recover the depth of each joint by predicting the root joint depth $d_{root} \in \mathbb{R}$. For j^{th} joint, the scaled pixel coordinate is converted to

$$\boldsymbol{p}_j^{\top} = \pi^{-1}(\mathcal{K}, (\boldsymbol{x}_j, d_j)^{\top} + (\boldsymbol{0}, d_{root})^{\top}) \tag{5}$$

The d_{root} is also scale-normalized to comply with the projection model. $\pi^{-1}(\cdot)$ is the inverse function of $\pi(\cdot)$. Till now, we have the joint locations in camera coordinate. Following [39], we train an IKNet to predict the quaternion of each joint, which is pose parameter θ.

For better convergence, we attach each transformation a loss function. $\mathcal{L}_{kpt^{2D}}$ is defined as the pixel-wise mean squared error (MSE) between the prediction and ground-truth $\mathcal{H}$. $\mathcal{L}_{kpt^{3D}}$ is the MSE between the prediction and ground-truth $\mathcal{P}$. $\mathcal{L}_d$ measures the MSE between the prediction and ground-truth d_{root}. The λs are the coefficients for each term.

$$\mathcal{L}_{\theta} = \lambda_{kpt^{2D}}\mathcal{L}_{kpt^{2D}} + \lambda_{kpt^{3D}}\mathcal{L}_{kpt^{3D}} + \lambda_d\mathcal{L}_d \tag{6}$$

To train IKNet, we adopt the same formulation as in [39]. Note that IKNet needs to be pretrained with $\mathcal{L}_{ik}$, but it also needs to be fine-tuned during the end-to-end training stage.

Shape-Branch. The shape-branch is to predict the shape parameter β, which takes $\mathcal{F}$, $\mathcal{H}$, and $\mathcal{D}$ as inputs via several ResNet layers [12]. Though we cannot directly supervise the β, the network can learn it from indirect supervision, such as re-projection to silhouette or depth, as discussed in Sec. 4.4. However, in practice, we find a simple regularization $\mathcal{L}_{\beta} = \|\beta\|^2$ is good enough for both training speed and final performance.

Mesh Formation. With θ and β, we can finally generate the mesh $\mathcal{M}$ through MANO layer [26]. The MANO layer cancels the translation, thus we need to add it back by translating the root location of $\mathcal{M}$ to $\boldsymbol{p}_{root} \in \mathcal{P}$, which is the 3D location of the root joint, to obtain $\hat{\mathcal{M}}$. The network should also be trained with a $\mathcal{L}_{mano}$, which is a function of both θ and β, measures the MSE between ground-truth and the predicted 3D joints $\mathcal{P}_{mano} \in \mathbb{R}^{k\times 3}$ extracted from $\mathcal{M}$.

Overall Loss Function. Then the overall loss function is

$$\mathcal{L} = \lambda_{\theta}\mathcal{L}_{\theta} + \lambda_{\beta}\mathcal{L}_{\beta} + \lambda_{mano}\mathcal{L}_{mano} \tag{7}$$

The λs are the coefficients for each term with corresponding subscripts.

3.4 Tailor Module

Current multi-stage pipelines for hand mesh reconstruction [33, 39], including ours, which usually construct the early stage for 2D information prediction and 3D information in the later stage, suffer from a precision decrease along the stages. A major reason behind this is that 2D information prediction is less ill-posed than 2D-3D mapping problem, which requires less but actually has more high-quality training samples. The *tailor module* is designed upon this observation to refine the hand mesh $\mathcal{M}$. It compares the output mesh to the intermediate representations from the multi-stage neural network. Since 2D keypoint is the most widely adopted representation and considered to be the most accurate one, we will discuss how to optimize with it. As for other intermediate representations, please refer to Sec. 4.4.

To fit the $\hat{\mathcal{M}}$ with the 2D image plane correctly and obtain a better hand mesh $\mathcal{M}^*$, we need to handle three constraints, namely hand scale, joint locations, and unnatural twist.

Scale Constraint. The predicted $\hat{\mathcal{M}}$ may appear inconsistent scale when re-projecting to the image, due to regression noise. We can optimize scale-compensate factor $s^* \in \mathbb{R}$ with an energy function, which leads to a more reasonable scale when projecting to the image plane.

$$\mathcal{E}_s(s^*) = ||\pi(\mathcal{K}; s^*\mathcal{P}_{mano}(\theta,\beta) + \boldsymbol{p}_{root}) - \mathcal{X}||_2^2 \tag{8}$$

Joint Location Constraint. As mentioned earlier, 2D keypoint estimation usually has higher accuracy. Thus when a well predicted $\mathcal{P}_{mano}$ is projected back to the image plane, it should be very close to the predicted $\mathcal{X}$.

$$\mathcal{E}_J(\theta,\beta) = ||\pi(\mathcal{K}; s^*\mathcal{P}_{mano}(\theta,\beta) + \boldsymbol{p}_{root}) - \mathcal{X}||_2^2 \tag{9}$$

Unnatural Twist Constraint. Since the losses are mostly joint location-oriented, it is very likely to cause a monster hand.We follow the design of [37] to repair such monster hand pose. Let $\boldsymbol{p}_a$, $\boldsymbol{p}_b$, $\boldsymbol{p}_c$, $\boldsymbol{p}_d \in \mathcal{P}_{mano}$ denote 4 joints of the fingers in tip to palm order. $\vec{V}_{ab}$ represents $\boldsymbol{p}_a - \boldsymbol{p}_b$, similar to $\vec{V}_{bc}$ and $\vec{V}_{cd}$. The energy function is

$$\mathcal{E}_g(\theta,\beta) = \|(\vec{V}_{ab} \times \vec{V}_{bc}) \cdot \vec{V}_{cd}\| - min(0, (\vec{V}_{ab} \times \vec{V}_{bc}) \cdot (\vec{V}_{bc} \times \vec{V}_{cd})) \tag{10}$$

Identity Constraint. We also have a regularization term to prevent the optimization from modifying the initialization too much. θ' and β' are the initial values of θ and β.

$$\mathcal{E}_{id}(\theta,\beta) = \|\beta - \beta'\|^2 + \|\theta - \theta'\|^2 \tag{11}$$

Two-Step Optimization. Eq. 8 and Eq. 9 are of the same formulation, while their optimization objectives are different. We find that jointly optimizing the energy function is unstable since scale constraint is a global constraint while the location constraint is a local constraint. Therefore we adopt a two-step optimization scheme, optimizing hand scale first and hand details later, which accelerates the optimization convergence.

Hand Scale Optimization. For s^*, Eq. 8 has an approximate analytical solution, we can obtain a near-optimal solution of s^* in 1 iteration.

$$s^* = z_{root} \frac{\sum_{i=1}^{k} (f_u x_i, f_v y_i)(u_i - u_{root}, v_i - v_{root})^\top}{\sum_{i=1}^{k} |(f_u x_i, f_v y_i)|^2} \tag{12}$$

Hand Detail Optimization. After obtaining a reasonable scale compensate factor, we can optimize the hand detail related energy function $\mathcal{E}$ in an iterative way.

$$\mathcal{E}(\theta,\beta) = \lambda_J \mathcal{E}_J(\theta,\beta) + \lambda_g \mathcal{E}_g(\theta,\beta) + \lambda_{id} \mathcal{E}_{id}(\theta,\beta) \tag{13}$$

The θ and the β are updated with the gradient function of $\mathcal{E}$ by an optimizer in each iteration. The λs are the coefficients for each term with corresponding subscripts.

3.5 HandTailor In-The-Wild without $\mathcal{K}$

The camera intrinsic $\mathcal{K}$ plays an important role in two critical points in the framework, transformation to camera coordinate (2D-3D mapping) and re-projection to 2D image plane (3D-2D mapping). However, sometimes $\mathcal{K}$ is unavailable, like an image from the internet, which limits the applicability of the framework. To address this HandTailor can be transited to weak perspective projection mode without any modification on network structure or weights.

Previous works for in-the-wild occasions usually treats the 2D-3D mapping by directly regressing 3D hand joints in camera coordinate [39] or regressing θ parameter [2, 37], and the 3D-2D mapping by estimating the s from neural networks [2, 37]. Such implicit treatments could cause interpretability and accuracy issues. Thanks to our multi-stage design, we find a way to calculate the scale factor s in the weak-perspective projection analytically. It is noteworthy that though s has the same meaning in both 2D-3D mapping and 3D-2D mapping, the estimation of s should be treated differently since the information available is not the same in these two phases.

2D-3D Mapping. In the 2D-3D mapping phase, we estimate the weak perspective projection scale factor s (see Eq.3) by utilizing the reference bone prior, which is set to a unit length.

$$[x_j, y_j, z_j]^\top = \Pi^{-1}(\mathcal{K}'; [u'_j, v'_j, d'_j]^\top) \tag{14}$$

Consider the bone length $(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2 = 1$, we can calculate s in $\mathcal{K}'$ by

$$s = \sqrt{\frac{\Delta u'^2 + \Delta v'^2}{1 - \Delta d'^2}} \tag{15}$$

$[\Delta x, \Delta y, \Delta z]$ and $[\Delta u', \Delta v', \Delta d']$ are the reference bone vectors in different coordinate systems.

3D-2D Mapping. As for the 3D-2D mapping, we can rely on more plausible cues to estimate s, and along with the translation $t \in \mathbb{R}^2$ on the plane as follows:

$$\mathcal{M}_{2D} = \Pi(\mathcal{M} * s) + t \tag{16}$$

We can directly calculate them based on the 2D keypoints we predicted in the early stage. t is the pixel coordinate of joint root $[u_{root}, v_{root}] \in \mathcal{X}$. s can be solved linearly by

$$s = \frac{\sum_{i=1}^{k} (x_i, y_i)(u_i - t_u, v_i - t_v)^\top}{\sum_{i=1}^{k} |(x_i, y_i)|^2} \tag{17}$$

Then the same network can directly process images without any camera information by slightly changing the computation scheme.

4 Experiments

4.1 Implementation

Hand Module. The input resolution of *hand module* is 256×256, and the intermediate resolutions are all 64×64. The training process is accomplished in a multi-step manner. We first train the keypoint estimation network for 100 epochs and IKNet for 50 epochs, finally train the whole network end-to-end for 100 epochs. We optimize the network through Adam with a learning rate of 3×10^{-4} and decrease to 3×10^{-5} in the end-to-end training stage. The network is trained with perspective projection. For in-the-wild occasions, we can directly change the computation scheme without any fine-tuning. The λs from Eq. 6 are set to 100, 1 and 0.1 respectively, and the λs from Eq. 7 are set to 1, 1 and 0.1 respectively.

Tailor Module. The *tailor module* is implemented on CPU with JAX [3], which can automatically differentiate the energy function, JIT compile and execute the optimization process. The scale-compensate factor is directly calculated via Eq. 12, and then optimize other constrains in an iterative manner to update β and θ. We utilize Adam with a learning rate of 0.003 as the optimizer, and the iteration number is set to 20 for the trade-off between accuracy and time cost. The λs from Eq. 13 are set to 1, 100 and 0.1 respectively.

4.2 Experiment Setting

We train and evaluate mainly on three datasets: Rendered Hand Dataset (RHD) [41], Stereo Hand Pose Tracking Benchmark (STB), and FreiHand dataset [42]. We report the percentage of correct keypoints (**PCK**), the area under the PCK curve (**AUC**), Procrustes aligned [12] mean per joint position error (**PA-MPJPE**), and Procrustes aligned mean per vertex position error (**PA-MPVPE**) as the main evaluation metrics. Note that previous works report AUC metric with a threshold range from 20*mm* to 50*mm*, denoted as AUC_{20-50}. According to [29], it is because sometimes the annotation errors from real datasets can exceed 10*mm*, and 20*mm* is agreeable to human judgment of two hands being close. However, as a synthetic dataset, RHD has no such problem. Thus to show the efficacy of *tailor module*, we also report the AUC from 5*mm* to 20*mm*, denoted as AUC_{5-20} on the RHD dataset.

4.3 Main Results

Comparison with SOTA Methods. In Tab. 1, we compare HandTailor with several previous state-of-the-art methods [2, 4, 7, 11, 18, 20, 35, 37, 41] on RHD, STB, and FreiHAND datasets. We can see that the proposed HandTailor can achieve state-of-the-art on RHD and STB benchmarks, and comparable results on FreiHAND with some mesh-convolution-based methods which involve extra mesh-level supervision, while HandTailor is a MANO-based method with only keypoint-level supervision. What's more, the architecture under weak perspective projection has only a little precision decay than perspective one, meaning that the weak perspective scheme can achieve reasonable simplification, and perspective one can have higher precision.

Method	RHD		STB
	$PCK_{20}\uparrow$	$AUC_{20-50}\uparrow$	
Z&B [41]	0.430	0.675	0.948
Boukhayma et al. [4]	0.790	0.926	0.995
Zhang et al. [42]	0.740	0.901	0.995
Ge et al.[11]	0.810	0.920	**0.998**
Yang et al.[35]	0.846	0.951	0.997
Hand module	0.833	0.949	0.997
HandTailor	**0.874**	**0.958**	**0.998**
HandTailor (w/o $\mathcal{K}$)	0.829	0.932	0.991

Method	FreiHAND	
	$PA-MPJPE\downarrow$	$PA-MPVPE\downarrow$
Boukhayma et al. [4]	35.0	13.2
MANO CNN [42]	11.0	10.9
YoutubeHand [18]	8.4	8.6
Pose2Mesh [2]	7.7	7.8
I2L-MeshNet [20]	**7.4**	**7.6**
Hand module	8.5	8.8
HandTailor	8.2	8.7
HandTailor (w/o $\mathcal{K}$)	8.9	9.2

Table 1: Comparison with state-of-the-art methods

Efficacy of Tailor Module. To better reflect the influence of *tailor module*, we select stricter metrics, PCK ranges from 5*mm* to 20*mm*. Also as we mentioned before, the *tailor module* only relates to the intermediate and final results provided by networks, so we conduct the plug-and-play experiments on several existing methods [2, 35, 39] with the demo models released by authors.

Method	Ours	[35]	[2]	[39]
Original	0.561	0.568	0.274	0.314
+ *Tailor module*	**0.658**	**0.601**	**0.341**	**0.356**

Table 2: A stricter metric AUC_{5-20} of previous work and the proposed HandTailor on RHD.

As shown in Tab. 2 and Fig. 3(a), the *tailor module* can bring a significant improvement, which proves its effectiveness in reducing errors and suiting many different networks. To

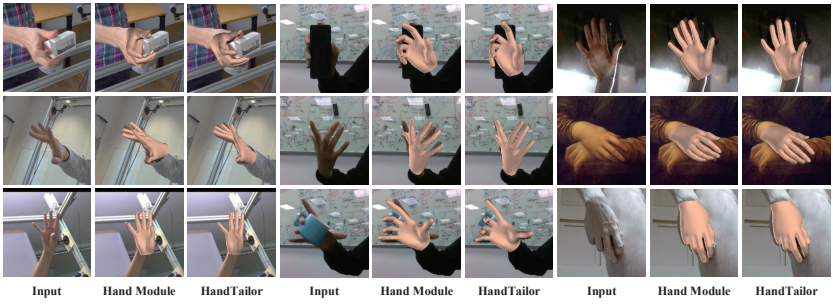
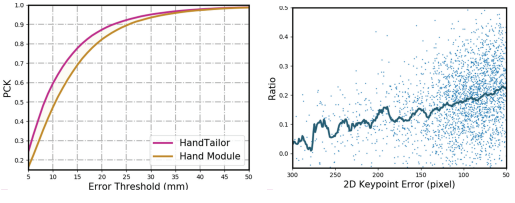


Figure 2: Qualitative result of HandTailor. There are three examples in each row. For each example, there are input image (left), output of *hand module* (middle), and the result of HandTailor (right). We can see that the *tailor module* improve the quality of the hand mesh remarkably. Our demonstrations include the samples from FreiHAND dataset [42], images captured by RealSense D435, and some pictures downloaded from the internet. The samples from FreiHAND and the internet are evaluated using weak-perspective projection, while samples from RealSense are evaluated using perspective projection.

show how the quality of intermediate results influences the performance of *tailor module*, we test and record the error of 2D keypoints and 3D joints of HandTailor on every sample in RHD. In Fig. 3(b), each point denote a sample in RHD dataset. The abscissa represents the 2D keypoint error of the sample, and the ordinate represents the ratio of 3D joints location error of hand module and the error eliminated by the tailor module. We can find that when the 2D keypoint estimation has higher precision, the *tailor module* can play a bigger role.



(a) (b)

Figure 3: (a): PCK curve with a threshold ranges from 5*mm* to 50*mm* on RHD. (b): The abscissa represents the error of 2D keypoint estimation of a specific sample on RHD, and the ordinate represents the influence of *tailor module*, which is the ratio of 3D joints location error of *hand module* and the error eliminated by the *tailor module*.

Speed of Tailor Module. Optimization-based methods tend to be slow. But the proposed *tailor module* can achieve high accuracy with a tiny time cost.We conduct the experiments with 20 iterations, which cost only 8.02*ms* per sample on PC with Intel i7-8700 CPU.

Iteration	10	20	50	100
Time (*ms*)	4.27	8.02	19.21	39.53
AUC_{20-50}	0.952	0.956	0.956	0.956
AUC_{5-20}	0.624	0.653	0.662	0.666

Table 3: The time cost and accuracy on RHD with different *tailor module* iteration numbers.

Qualitatively Result. It can be seen from Fig. 2 that when we only rely on the *hand module*, the hand skeleton seems correct, but once it is re-projected onto the original image, there will

be numerous evident small errors, such as a few degrees of finger deviation. Once we utilize the *tailor module* to fine-tune the hand reconstruction results, these errors can be largely fixed, and we can make the projection more coherent to the image.

4.4 Ablation Study

In this part, we evaluate some key components of our approach on RHD.

Losses of Shape-Branch. To supervise the shape-branch of *hand module*, despite the regularization loss $\mathcal{L}_\beta$ mentioned before, a depth loss $\mathcal{L}_{\beta^D}$ and a silhouette loss $\mathcal{L}_{\beta^S}$ can also be conducted through a differential renderer [16]. $\mathcal{L}_{\beta^D}$ and $\mathcal{L}_{\beta^S}$ are the MSE losses between predicted and ground-truth depth and silhouette. Tab. 4 shows that these losses cannot enhance the performance effectively. So we only use a simple regularization $\mathcal{L}_\beta$ while training.

Energy Function. The energy function of *tailor module* has three components affecting the performance on PCK metric: $\mathcal{E}_J$, $\mathcal{E}_g$, and $\mathcal{E}_{id}$.Besides these terms, the *tailor module* can also utilize depth constraint $\mathcal{E}_d$ that measures the distance between rendered and input depth, and silhouette constraint $\mathcal{E}_s$ that measures the distance between rendered and input silhouette. The rendered depth and silhouette are generated via a differential renderer, and the input depth and silhouette are both extracted by watershed from the depth map.

$\mathcal{L}_\beta$	$\mathcal{L}_{\beta^D}$	$\mathcal{L}_{\beta^S}$	AUC_{20-50}	AUC_{5-20}
✓			0.958	0.658
✓	✓		0.956	0.653
✓		✓	0.955	0.651

$\mathcal{E}_g$	$\mathcal{E}_{id}$	$\mathcal{E}_d$	$\mathcal{E}_s$	AUC_{20-50}	AUC_{5-20}	Time(ms)
✓	✓			0.958	0.658	8.02
✓				0.958	0.657	8.01
	✓			0.958	0.658	7.98
✓	✓	✓		0.950	0.644	42.11
✓	✓		✓	0.948	0.640	42.08

Table 4: Ablation study on the loss functions of shape-branch (left), and how different constraints of *tailor module*. influence the precision and speed (right).

We can see from Tab. 4 that $\mathcal{E}_g$ and $\mathcal{E}_{id}$ do not improve the precision, but they do affect the hand mesh quality as shown in Fig. 4. Also, depth and silhouette constraints cannot improve *tailor module*. This is because they make the energy function more complex and hard to converge. These constraints also bring a heavy overhead to *tailor module*.

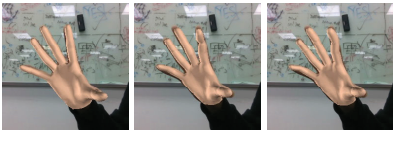


Figure 4: From left to right are the output of the full *tailor module*, the output of the *tailor module* without unnatural twist constraint, and the output without identity constraint.

5 Conclusion

In this paper, we propose a novel framework HandTailor for monocular RGB 3D hand recovery, combining a learning-based *hand module* and an optimization-based *tailor module*. We can adapt both perspective projection and weak perspective projection for high precision and in-the-wild scenarios. And the *tailor module* can bring significant improvement for the whole pipeline both qualitatively and quantitatively. In the future, we will try to solve the hand reconstruction task when the hand is holding objects or two hands are interacting.

References

[1] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1067–1076, 2019.

[2] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 3d hand shape and pose from images in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10843–10852, 2019.

[3] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[4] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–682, 2018.

[5] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[6] Yujin Chen, Zhigang Tu, Di Kang, Linchao Bao, Ying Zhang, Xuefei Zhe, Ruizhi Chen, and Junsong Yuan. Model-based 3d hand reconstruction via self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10451–10460, 2021.

[7] Hongsuk Choi, Gyeongsik Moon, and Kyoung Mu Lee. Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose. In *European Conference on Computer Vision*, pages 769–787. Springer, 2020.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[9] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3593–3601, 2016.

[10] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. Hand pointnet: 3d hand pose estimation using point sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8417–8426, 2018.

[11] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 3d hand shape and pose estimation from a single rgb image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 10833–10842, 2019.

[12] John C Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.

[13] Jiajun Gu, Zhiyong Wang, Wanli Ouyang, Jiafeng Li, Li Zhuo, et al. 3d hand pose estimation with disentangled cross-modal latent space. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 391–400, 2020.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5 d heatmap regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 118–134, 2018.

[16] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[17] Dominik Kulon, Haoyang Wang, Riza Alp Güler, Michael Bronstein, and Stefanos Zafeiriou. Single image 3d hand reconstruction with mesh convolutions. *arXiv preprint arXiv:1905.01326*, 2019.

[18] Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael M Bronstein, and Stefanos Zafeiriou. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4990–5000, 2020.

[19] Jameel Malik, Ibrahim Abdelaziz, Ahmed Elhayek, Soshi Shimada, Sk Aziz Ali, Vladislav Golyanik, Christian Theobalt, and Didier Stricker. Handvoxnet: Deep voxel-based network for 3d hand shape and pose estimation from a single depth map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7113–7122, 2020.

[20] Gyeongsik Moon and Kyoung Mu Lee. I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 752–768. Springer, 2020.

[21] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1284–1293, 2017.

[22] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018.

[23] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[24] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BmVC*, volume 1, page 3, 2011.

[25] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. Using a single rgb frame for real time 3d hand pose estimation in the wild. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 436–445. IEEE, 2018.

[26] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (ToG)*, 36(6): 245, 2017.

[27] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen 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*, pages 3633–3642, 2015.

[28] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. Cross-modal deep variational hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–98, 2018.

[29] James S Supancic, Grégory Rogez, Yi Yang, Jamie Shotton, and Deva Ramanan. Depth-based hand pose estimation: data, methods, and challenges. In *Proceedings of the IEEE international conference on computer vision*, pages 1868–1876, 2015.

[30] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-icp for real-time hand tracking. In *Computer Graphics Forum*, volume 34, pages 101–114. Wiley Online Library, 2015.

[31] Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *ACM Transactions on Graphics (ToG)*, 35(6):1–11, 2016.

[32] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Dual grid net: hand mesh vertex regression from single depth maps. In *European Conference on Computer Vision*, pages 442–459. Springer, 2020.

[33] Linlin Yang and Angela Yao. Disentangling latent hands for image synthesis and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9877–9886, 2019.

[34] Linlin Yang, Shile Li, Dongheui Lee, and Angela Yao. Aligning latent spaces for 3d hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2335–2343, 2019.

[35] Lixin Yang, Jiasen Li, Wenqiang Xu, Yiqun Diao, and Cewu Lu. Bihand: Recovering hand mesh with multi-stage bisected hourglass networks. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020.

[36] Shanxin Yuan, Guillermo Garcia-Hernando, Björn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, et al. Depth-based 3d hand pose estimation: From current achievements to future goals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2018.

[37] Xiong Zhang, Qiang Li, Hong Mo, Wenbo Zhang, and Wen Zheng. End-to-end hand mesh recovery from a monocular rgb image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2354–2364, 2019.

[38] Long Zhao, Xi Peng, Yuxiao Chen, Mubbasir Kapadia, and Dimitris N Metaxas. Knowledge as priors: Cross-modal knowledge generalization for datasets without superior knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6528–6537, 2020.

[39] Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. Monocular real-time hand shape and motion capture using multi-modal data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5346–5355, 2020.

[40] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[41] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *Proceedings of the IEEE international conference on computer vision*, pages 4903–4911, 2017.

[42] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 813–822, 2019.

[43] Christian Zimmermann, Max Argus, and Thomas Brox. Contrastive representation learning for hand shape estimation. *arXiv preprint arXiv:2106.04324*, 2021.