3D Hand Reconstruction with Novel Parametric Hand Model and Render-and-Compare

Zhuoran Zhao
School of Computing
National University of Singapore
Singapore, 117417
zhuoran.zhao@u.nus.edu

Abstract

Reconstructing a 3D hand from a single-view RGB image is a challenging task in computer vision. Not only because obtaining 3D hand annotations is expensive, but also because heavy occlusions and self-similarity are common in human hands. Although most prior studies leverage parametric hand models as a strong prior to realize robust reconstruction, most existing models only model the skeleton and mesh of hands, which cannot generate 3D hand to a new level of realism. In this work, I propose a pipeline to reconstruct a realistic 3D hand from monocular images with a novel parametric hand model NIMBLE, which can generate hand mesh with fine-grained texture. I further integrate a render-and-compare branch to minimize differences between the input image and rendered output with a differentiable renderer. I also take a step towards investigating the effect of different learning schemes, confidence scores on noisy labels and different 3D losses. Finally, I make a comparison between MANO and NIMBLE. Code is publicly available at https://github.com/delaprada/3D_hand_reconstruction.

1 Introduction

Reconstructing 3D hand from monocular images is a long standing problem for computer vision tasks [6, 28, 23], which has many applications such as: augmented reality (AR/VR) and human-machine interaction. Realistic 3D hand reconstruction also facilitates recent applications such as Metaverse and AI Generated Content (AIGC). Many conventional studies rely on using RGB-D sensor [30] or multi-view setups [26] to reconstruct 3D hand. However, with the emergence of deep learning, RGB images as a single input modality becomes accessible [34, 17, 3, 4]. But there still remains many challenges to recover 3D hand pose and shape from monocular images because of ambiguities in depth and scale. To solve these problems, many model-based methods [2, 3, 32] make use of parametric hand models to approximate hand shape, while other model-free methods [13, 9] use graph CNNs to directly regress vertex positions.

However, most model-based methods use parametric models which are relatively coarse and only focus on modeling the skeleton of the hand without considering muscles, bones, skins and wrist. For example, the underlying geometry of MANO [22] only has 778 vertices and limited expressivity of hand shape. Moreover, existing parametric models are too ideal to represent the complexity and diversity of real hands. The texture of most generated hand mesh is not fine-grained enough, either by directly adding RGB value to each vertice [6], or by unwrapping the hand mesh to map colors from the image to the texture map [24]. However, realistic hands are often rich in appearance, such as color of skins, wrinkles, palm prints.

Lack of high quality labeled training data of hand poses restricts the development of 3D hand pose estimation and reconstruction. Labeling real-world data to an accurate degree is

labor-intensive and requires multi-view setups. To further push the limit of data-driven methods, researchers start to shift their focus from manual labeling on collected images [31, 35, 16] to large-scale synthesizing [17, 34, 7]. Synthesizing hand data has many benefits over manual labeling, which not only guarantees rich ground-truth labels with low cost, but also autonomously diversifies synthesized data with different poses and backgrounds.

Render-and-compare scheme is a novel method to bridge the gap between 2D and 3D by allowing 2D image pixels to be related back to 3D properties of the hand shape, such as the positions of mesh vertices, enabling 3D shapes to be learnt without explicit 3D supervision and only image-level supervision. However, current methods using render-and-compare scheme as one of the training losses shows little improvement to the overall accuracy compared with other losses [6, 21]. I believe its potential remains relatively unexplored.

Motivated by the above observations, this works seeks to propose a pipeline to reconstruct a realistic 3D hand from monocular images with a novel parametric hand model, NIMBLE, which is a non-rigid parametric hand model including bones, muscles and high-quality texture. I further incorporate a render-and-compare branch into the pipeline to minimize 3D reconstruction errors as well as discrepancies between inputs and rendered images from estimated outputs. I also aim to conduct experiments to evaluate the impact of different training schemes and factors of the pipeline.

Due to the time limit, some works may be further researched in the future:

1. Firstly, motivated by DART [7], which extends MANO with not only rich texture and articulated wrist, but also diverse accessories, extending the existing parametric hand models with hand-object interaction is also important, since most methods can not perform well in such a scenario. Existing multi-view hand datasets with hand-object interaction contains noisy labels and the variety of manipulated objects is relatively small, which prevents the model from realizing robust estimation and reconstruction. More synthesized data with accurate ground-truth labels in hand-objection interaction can alleviate this problem.
2. Secondly, model-based methods are greatly restricted by the representation power of the underlying hand model and cannot represent the whole diversity of hand shapes. Some model-free methods have shown their competence in 3D hand reconstruction task. A hybrid of these two methods may contribute to a better result [24].
3. Since most render-and-compare schemes applied in 3D hand reconstruction show little improvement to the accuracy, more experiments can be done to test which render-and-compare schemes are the best to apply to 3D hand reconstruction tasks, such as photorealistic loss, silhouette loss, depth map loss, etc. Moreover, camera intrinsic parameters like focal length also affect the rendering, but most studies only consider rotation, translation and scale information. To better represent the hand joints and mesh in camera coordinate system, camera intrinsic parameters can be considered in the training process [18].

2 Related Work

2.1 3D hand pose and shape reconstruction

While 2D hand pose estimation only needs to estimate 2D keypoints, 3D hand reconstruction is more complex. Generally, 3D hand reconstruction can be categorized into Image-to-Pose and Image-to-Shape depending on the representation of articulated geometry. Image-to-Pose only recovers the sparse hand joints' locations while Image-to-Shape recovers the surface geometry of the hand, which has richer information of hand pose and shape. Most previous works of Image-to-Shape can be divided into two groups: model-free methods for mesh vertex regression and model-based methods for parameter regression.

For model-free methods, graph CNNs are widely adopted by some works for 3D hand reconstruction since mesh can be considered as graph. These works directly regress vertex positions with gradual refinement and some of them make use of spectral [12, 8] or spiral operator [13, 5] to process mesh vertices. Model-free methods can achieve accurate reconstruction results and are robust against abnormality.

Table 1: Comparison of Neural 3D Mesh Renderer and Pytorch3D

	Neural 3D Mesh Renderer [10]	Pytorch3D [20]
Differentiable	✔	✔
Modular	✗	✔
Batching	homogeneous	heterogeneous
Rendered Image Type	RGB	RGBA
Silhouette Image	Obtain directly	Obtain from alpha channel
Camera Intrinsic Matrix	$3 * 3$	$4 * 4$
Community	Inactive	Active

For model-based methods, MANO [22] is one of the most widely used parametric hand model. MANO models hand mesh with 778 vertices, which are driven by pose and shape parameters. But MANO is limited in its expressiveness and lacks real biomechanical constrains. HTML [19] uses principal component analysis (PCA) to build a textured parametric hand model, which captures textures with external factors like lighting, shading and materials. NIMBLE [14] brings 3D hand model into a new level of realism, with bones, muscles and skins. NIMBLE provides diffuse, normal, and specular maps separated from external factors, which is more suitable for photometric rendering pipeline. DART [7] is a new parametric hand model that not only considers texture modeling, but also considers articulated wrist and daily accessories. It also adds common traits of hand inside the texture, such as moles, nail colors, scars, palm prints.

In this work, I employ NIMBLE as the parametric hand model in the pipeline that maps pose and shape parameters to a fine-grained hand mesh.

2.2 Differentiable rendering

Differentiable rendering is a novel research area in computer vision, which bridge the gap between 2D and 3D. It projects 3D data to 2D images and 3D predictions can be made using only image-level supervision. Inverting the rendering step can be helpful to relate the 2D pixel-level loss back to 3D shape properties such as the positions of mesh vertices. Differentiable rendering enables unsupervised learning and reduces the requirement of 3D data collection and annotation. Neural 3D Mesh Renderer [10] is the first general-purpose neural mesh renderer based on deep learning, which approximates the rasterization gradient with a manual-design linear function. However, the proposed linear function can not fully model the nonlinearity of real rasterizers and the forward and backward propagations remain inconsistency. Soft rasterizer proposed in [15] can provide an accurate soft approximation of the standard rasterizer and estimate rendering derivative with higher accuracy. Pytorch3D [20] is a more modular, efficient and scalable rendering engine compared with [10, 15], which achieves modularity by decomposing the rendering pipeline into stages and supports heterogeneous batching.

Here I try using two renderer engines, Neural 3D Mesh Renderer and Pytorch3D to render hand mesh for achieving render-and-compare. Comparison of their differences is shown in table 1.

3 Methodology

My proposed pipeline is illustrated in Figure 1. In particular, I first use an encoder that receives an image of a hand as input and outputs the pose, shape, appearance and camera viewpoint (Section 3.1). Then, I use a model-based hand decoder to generate a textured 3D hand (Section 3.2). I render this 3D hand into 2D images (Section 3.4) and design a photometric consistency loss to perform render-and-compare (Section 3.5). I will describe my methods in detail in the following sections.

3.1 Deep Hand Encoder

A deep hand encoder is composed of two components: a backbone and regressors. Backbone is used to extract features from the input image and regressors are used to encode the image into

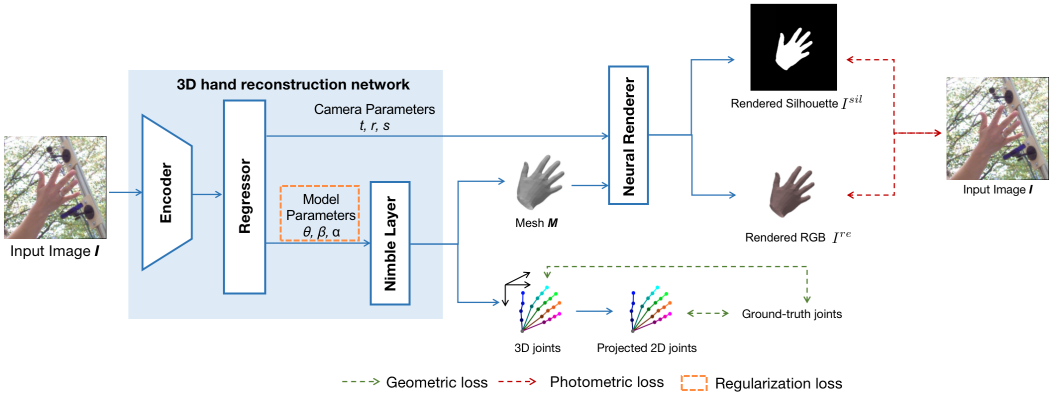

Figure 1: Overview of the proposed pipeline. Encoder is used to extract image features and regressors is used to predict parameters for nimble layer and neural renderer. I adopt 3 types of losses to train the pipeline: geometric loss, photometric loss and regularization loss, where geometric loss includes joint loss and bone orientation loss.

parametric model parameters and camera parameters. MANO parameters $m = (\beta, \theta)$ consists of pose parameter $\theta \in \mathbb{R}^{30}$, shape parameter $\beta \in \mathbb{R}^{10}$. NIMBLE parameters $n = (\beta, \theta, \alpha)$ consists of pose parameter $\theta \in \mathbb{R}^{30}$, shape parameter $\beta \in \mathbb{R}^{20}$, and appearance parameter $\alpha \in \mathbb{R}^{10}$. Camera parameters $c = (t, R, s)$ consists of translation parameter $t \in \mathbb{R}^3$, rotation parameter $R \in \mathbb{R}^3$, and scale factor $s \in \mathbb{R}^1$, which allows to project a posed 3D hand back onto a 2D image.

3.2 Model-based Hand Decoder

Here I try using two parametric hand models MANO and NIMBLE to reconstruct 3D hand mesh.

MANO takes pose θ, shape β as inputs and returns corresponding 3D joints $\in \mathbb{R}^{21\times3}$ and vertices $\in \mathbb{R}^{778\times3}$ of the generated hand mesh. The pose of hand mesh generally has two types of representations: PCA and axis angle. The shape of hand mesh only has one representation: PCA. Here, pose is determined through 30 PCA components and shape is determined through 10 PCA components that encode the captured hand pose and shape variance of the MANO hand scans [22].

NIMBLE has some differences with MANO. NIMBLE takes pose θ, shape β and appearance α as inputs and returns skin vertices $\in \mathbb{R}^{5990\times3}$, muscle vertices$\in \mathbb{R}^{5635\times3}$, bone vertices$\in \mathbb{R}^{3345\times3}$, bone joints$\in \mathbb{R}^{25\times3}$ and texture image$\in \mathbb{R}^{1024\times1024\times9}$. Hand mesh generated by NIMBLE is defined with 25 anatomical joints, so some transformation is required in order to perform training. Here, a transformative function provided by NIMBLE layer is used to transform NIMBLE skin vertices to MANO vertices. Then a MANO joint regressor is used to regress 16 joints from MANO vertices. As the hand skeleton in MANO does not contain finger tip joints, I append the hand skeleton with 5 vertices from hand mesh that correspond to these keypoints. The final 3D joint output of NIMBLE counts 21 keypoints.

3.3 Camera model

In order to represent the output joints and mesh $J(\beta, \theta)$ and $M(\beta, \theta)$ in camera coordinate system, predicted scale s, rotation R and translation t should be applied to the original joints J_0 and mesh M_0. The final representation of the hand mesh M and joint J are:

$$M = sM_0R + t$$
$$J = sJ_0R + t$$

Also, additional orthographic projection Π is required to re-project the 3D joints into 2D image plane:

$$J_{2d} = s\Pi(J_0R) + t$$

3.4 Image Formation

A neural renderer [10, 20] is used to transform the 3D hand mesh into 2D images. In order to render textured hand mesh, texture image generated by NIMBLE is incorporated in the rendering stage. Here I use the Pytorch3D TexturesUV method Γ to incorporate texture image I^{tex}, UV coordinates of faces f and UV coordinates of vertices v to render textured hand mesh. Final texture representation T is:

$$T = \Gamma(I^{tex}, f, v)$$

The neural renderer Δ takes the 3D hand mesh M, camera parameters c and texture information T as inputs and generate a silhouette I^{sil} and an RGB image I^{re} of the 3D hand: $I^{sil}, I^{re} = \Delta(M, c, T)$.

3.5 Training Objective

I combine multiple losses to train my pipeline, which includes joint loss, bone orientation loss, photometric loss and parameter regularization loss:

$$L = w_{joint}L_{joint} + w_{bone}L_{bone} + w_{photo}L_{photo} + w_{reg}L_{reg}$$

Joint loss For joint loss, I use 3D joint loss L_{3d}, 2D joint loss L_{2d} or normalized 3D joint loss L_{3d}^{norm} for training:

$$L_{joint} = w_{2d}L_{2d} + w_{3d}L_{3d} + w_{3d_norm}L_{3d}^{norm}$$

3D joint loss directly uses ground-truth 3D joints J_{gt} as supervision while 2D joint loss uses 2D joints J_{gt}^{pro} which are projected from 3D ground-truth joints. Apart from directly using ground-truth 3D joints, I also try using normalized ground-truth 3D joints J_{gt}^{norm} to train, which calculate 3D joint error in hand-centric coordinate. Their representations are as follows:

$$L_{3d} = \|J_{gt} - J_{pred}\|_2^2$$

$$L_{2d} = \|J_{gt}^{pro} - J_{pred}^{pro}\|_2^2$$

$$L_{3d}^{norm} = \|J_{gt}^{norm} - J_{pred}^{norm}\|_2^2$$

where J_{pred} refers to the 3D joint coordinates predicted by the model, J_{pred}^{pro} refers to 2D joint coordinates projected from J_{pred}, J_{pred}^{norm} refers to normalized J_{pred}.

Bone orientation loss Bone orientation loss L_{bone} ensures bones of two sets of keypoints to be aligned:

$$L_{bone} = \|b_{gt} - b_{pred}\|_2^2$$

where b_{gt} and b_{pro} are the normalized bone vectors of ground-truth 2D joints and projected 2D joints.

Photometric loss Photometric loss L_{photo} calculates the discrepancies between inputs and rendered outputs to perform render-and-compare, which consists of two terms: L_{rgb} and L_{sil}:

$$L_{photo} = L_{rgb} + L_{sil}$$

The first term L_{rgb} refers to the difference between input image and rendered RGB image, which is computed by averaging the least absolute deviation distance for all pixels and the structural similarity (SSIM) loss between these two images:

$$L_{rgb} = \frac{1}{H_{re} \times W_{re}} \sum_{(u,v) \in H_{re} \times W_{re}} \|I_{u,v} - I_{u,v}^{re}\|_2 + (1 - SSIM(I, I^{re}))$$

where $H_{re} \times W_{re}$ denotes the output resolution of I^{re}; $I_{u,v}$ and $I^{re}_{u,v}$ denote the color of pixel (u, v) in input image I and rendered RGB image I^{re}.

The second term L_{sil} refers to the difference between input image and rendered silhouette, which is computed by calculating SSIM loss between input image I and rendered silhouette I^{sil}:

$$L_{sil} = 1 - SSIM(I, I^{sil})$$

Regularization loss Regularization loss L_{reg} is applied to model parameters to penalize implausible 3D hand pose and shape to make sure the reconstructed hand is undistorted. For MANO hand model, L2 regularizers are added to pose and shape parameters. For NIMBLE hand model, L2 regularizers are added to pose, shape and appearance parameters.

$$L_{reg} = w_{pose}\|\theta\|_2^2 + w_{shape}\|\beta\|_2^2 + w_{ap}\|\alpha\|_2^2$$

4 Experiments

In this section, I first introduce dataset, evaluation metrics and implementation details. Then I show the results of my approach and conduct some comparisons and analysis.

4.1 Dataset

The proposed pipeline is trained on FreiHAND dataset. FreiHAND dataset [35] is a large-scale, multi-view hand dataset that is accompanied by both 3D hand pose and shape annotations, which contains 32,560 training samples and 3,960 test samples.

4.2 Evaluation Metrics

I evaluate my approach by calculating the errors of 3D joints and 3D vertices. I compute the mean per joint position error (MPJPE) for 3D joints and mean per vertex position error (MPVPE) for 3D meshes in cm. I also calculate the area under curve (AUC) for joints (AUC_J) and vertices (AUC_V) in the euclidean space for all joints and vertices in all images in cm. I also calculate the F-score [11], which is a harmonic mean of precision and recall at given threshold. I report the threshold at 5mm and 15mm as F_5 and F_{15}. Predictions are compared with procrustes alignment.

4.3 Implementation

The proposed pipeline is implemented by PyTorch. For 3D reconstruction network, the EfficientNet [27] pretrained on the ImageNet dataset is used as backbone. The optimization is carried out using Adam with batch size of 64. The number of training epochs is 180. The initial learning rate is 10^{-3} and reduced by a factor 2 after every 30 epochs. The experiments are conducted on one NVIDIA RTX A5000 GPU. For more details, please refer to the attached code.

4.4 Comparison with State-of-the-art Methods

I first use MANO as the parametric hand model in the pipeline and train it with 2D joint loss, bone orientation loss, photometric loss and regularization loss, which is a weakly-supervised scheme. Since my pipeline refers to S2HAND [6], I make a comparison on FreiHAND with it (see Table 2). Note that my pipeline uses the same backbone, parametric hand model, render-and-compare strategy and similar regularization method as S2HAND. The only difference is on the training scheme, where S2HAND uses OpenPose 2D detected keypoints as supervision and my pipeline uses ground-truth 2D keypoints as supervision. The results shows that using weakly-supervised scheme underperforms using self-supervised scheme. I think it is because of the confidence scores. In S2HAND, confidence scores are additionally embedded into the joint loss and bone orientation loss. Confidence scores of OpenPose is a number between 0 and 1 and shows the probability of keypoints being detected correctly. These confidence information is additional correction of OpenPose labels since they are not ground-truth and have many incorrect predictions. If every detected labels are treated equally, model may overfit to the noisy and inaccurate labels, resulting in poor

Table 2: Comparison of main results on FreiHAND test set

Supervision	Method	Backbone	$AUC_J \uparrow$	MPJPE $\downarrow$	$AUC_V \uparrow$	MPVPE $\downarrow$	$F_5 \uparrow$	$F_{15} \uparrow$
OpenPose	S2HAND	EfficientNet	0.77	1.18	0.77	1.19	0.48	0.92
2D	Ours	EfficientNet	0.74	1.29	0.74	1.30	0.43	0.90

Table 3: Comparison of results with different supervised bone orientation loss

Method	$AUC_J \uparrow$	MPJPE $\downarrow$	$AUC_V \uparrow$	MPVPE $\downarrow$	$F_5 \uparrow$	$F_{15} \uparrow$
WSL w/o confidence	0.74	1.33	0.74	1.33	0.43	0.89
WSL w confidence	0.73	1.38	0.72	1.40	0.41	0.88
SSL w/o confidence	0.74	1.33	0.74	1.33	0.42	0.90
SSL w confidence	**0.74**	**1.29**	**0.74**	**1.30**	**0.43**	**0.90**

performance on unseen data. With confidence scores, model can discriminate which labels should be paid more attention to and discount the influence from those incorrect labels. Confidence information is a widely used techniques to rectify pseudo labels in semi-supervised learning [1, 25, 29, 33].

Here, I take a step further to test my assumption whether it is the confidence score that contributes to these results. I add confidence score to the bone orientation loss in weakly-supervised scheme and remove confidence score from bone orientation loss in self-supervised scheme. Results are shown in Table 3 and Fig. 2. It shows that self-supervised bone orientation loss with confidence score performs the best and adding confidence score to weakly-supervised bone orientation loss worsens the results. It proves that only when confidence score is combined with noisy labels can improve the results. Weakly-supervised scheme is worse than self-supervised scheme because it heavily relies on less accurate and less informative labels without uncertainty weighting. Although weakly-supervised scheme uses ground-truth 2D keypoints projected from ground-truth 3D keypoints, there are still some inaccurate labels. Such a situation occurs when the annotators are careless or keypoints are difficult to be positioned. This can lead to lower accuracy and robustness in the model. Adding OpenPose confidence score to weakly-supervised scheme is detrimental since it may increase the probability of inaccurate labels, thus perturbing the model.

4.5 Comparison of using different GT 3D as supervision

3D ground-truth joints can be regarded as a powerful supervision. I try using two types of 3D supervisions: 3D joint loss and normalized 3D joint loss and conducting further experiments to compare their impacts on the results. As shown in table 4 and 5, adding a ground-truth 3D joint loss

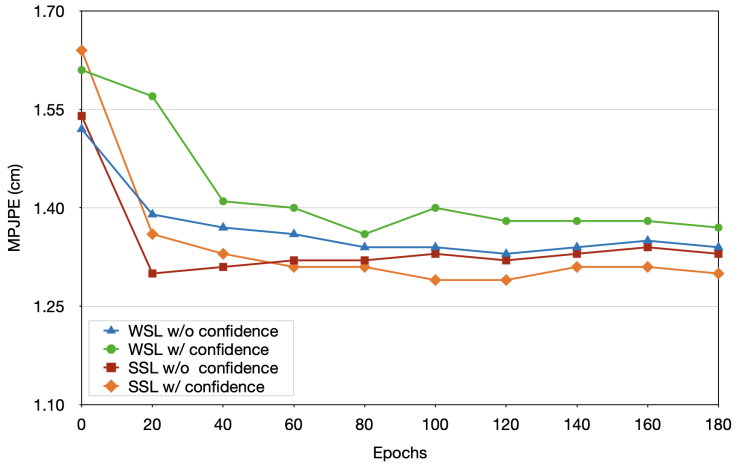

Figure 2: Comparison with different supervised bone orientation loss on different epochs.

Table 4: Comparison of results adding different 3D losses with procrustes alignment

Supervision	Method	$AUC_J \uparrow$	MPJPE $\downarrow$	$AUC_V \uparrow$	MPVPE $\downarrow$	$F_5 \uparrow$	$F_{15} \uparrow$
2D	Ours	0.74	1.33	0.74	1.33	0.42	0.89
+ 3D	Ours	0.73	1.39	0.73	1.38	0.41	0.89
+ norm 3D	Ours	**0.76**	**1.24**	**0.76**	**1.22**	**0.46**	**0.91**

Table 5: Comparison of results adding different 3D losses

Supervision	Method	$AUC_J \uparrow$	MPJPE $\downarrow$	$AUC_V \uparrow$	MPVPE $\downarrow$	$F_5 \uparrow$	$F_{15} \uparrow$
2D	Ours	0.11	11.42	0.11	11.45	0.06	0.22
+ 3D	Ours	0.13	8.31	0.13	8.40	0.08	0.27
+ norm 3D	Ours	**0.15**	**7.70**	**0.15**	**7.73**	**0.08**	**0.29**

does not improve aligned results, but rather unaligned results while adding a normalized ground-truth 3D joint loss can improve both aligned and unaligned results. I think the reason why adding 3D joint loss does not improve the aligned results is that the model needs to additionally learn the accurate position and scale of generated 3D hand since unaligned results are improved. When using normalized 3D joint loss, the coordinate system is attached to one of the joints and joint loss is calculated in object coordinate system. As a result, there is no need for model to learn additional translation, scale and rotation information. Moreover, using procrustes alignment to evaluate 3D hand pose and shape incorporates additional steps to translate, scale and rotate the keypoints before calculating mean per-joint position error. Therefore, training with normalized 3D joint loss can provide a better aligned result. On the other hand, I think the reason why adding normalized 3D joint loss can improve unaligned result is because the 2D projected keypoints helps model to learn some translation, scale and rotation information.

4.6 Comparison of MANO and NIMBLE

I use two different hand models MANO and NIMBLE to reconstruct 3D hand here, the results is shown in table 6. Both of them use normalized 3D joint loss and regularization loss as as supervision. Though NIMBLE does not outperform MANO due to the fundamental difference of joint definition, it can achieve a comparable quantitative result and predict photorealistic 3D hand (Fig. 4 and Fig. 5).

4.7 Qualitative results

Qualitative results with MANO and NIMBLE are shown in Fig. 3, Fig. 4 and Fig. 5.

5 Conclusions

In this project, I build up a pipeline for 3D hand reconstruction from monocular images with parametric hand models and perform render-and-compare with differentiable renderers. I further investigate the impact of different training strategies on experimental results, such as different learning schemes, confidence score of noisy labels and different types of 3D losses.

Table 6: Comparison of results using different hand models

Hand Model	Method	$AUC_J \uparrow$	MPJPE $\downarrow$
MANO	Ours	0.76	1.24
NIMBLE	Ours	0.68	1.60

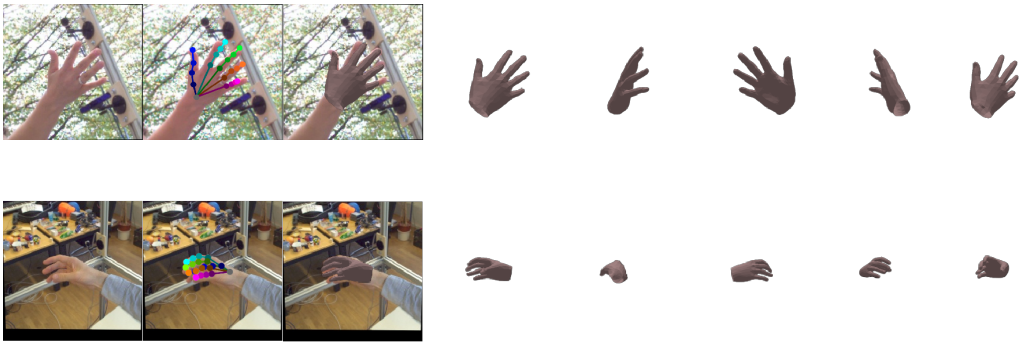

Figure 3: Qualitative results on FreiHAND test set with MANO.

Figure 4: Qualitative results on FreiHAND test set with NIMBLE.

6 Discussion

In the process of building up this pipeline, there still have some unsolved problems that needs to be further investigated. First of all, rendering with Pytorch3D requires camera intrinsic matrix to be $4 * 4$ while commonly used is $3 * 3$. Due to not detailed enough technical documents and inactive community, this problem still remains unsolved and camera intrinsic matrix information can not be integrated into Pytorch3D differentiable renderer. Secondly, weighting factors of regularization loss has a significant impact on the training result with NIMBLE hand model. Although large weighting factors can ensure plausible hand mesh, pipeline can be difficult to converge. Conversely, small weighting factors can lead to severe hand collision problem. Therefore, proper weighting factors for the L2 regularizers need to be further examined.

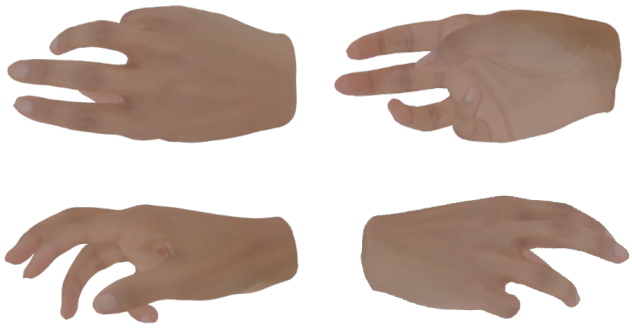

Figure 5: Detailed representative results on FreiHAND test set with NIMBLE.

References

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[2] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 1067–1076, 2019.

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

[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] Xingyu Chen, Yufeng Liu, Chongyang Ma, Jianlong Chang, Huayan Wang, Tian Chen, Xiaoyan Guo, Pengfei Wan, and Wen Zheng. Camera-space hand mesh recovery via semantic aggregation and adaptive 2d-1d registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13274–13283, 2021.

[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] Daiheng Gao, Yuliang Xiu, Kailin Li, Lixin Yang, Feng Wang, Peng Zhang, Bang Zhang, Cewu Lu, and Ping Tan. Dart: Articulated hand model with diverse accessories and rich textures. *arXiv preprint arXiv:2210.07650*, 2022.

[8] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 10833–10842, 2019.

[9] Shaoxiang Guo, Eric Rigall, Lin Qi, Xinghui Dong, Haiyan Li, and Junyu Dong. Graph-based cnns with self-supervised module for 3d hand pose estimation from monocular rgb. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(4):1514–1525, 2020.

[10] 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.

[11] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[12] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4501–4510, 2019.

[13] 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.

[14] Yuwei Li, Longwen Zhang, Zesong Qiu, Yingwenqi Jiang, Nianyi Li, Yuexin Ma, Yuyao Zhang, Lan Xu, and Jingyi Yu. Nimble: a non-rigid hand model with bones and muscles. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022.

[15] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. *arXiv preprint arXiv:1901.05567*, 2019.

[16] Gyeongsik Moon, Shoou-I Yu, He Wen, Takaaki Shiratori, and Kyoung Mu Lee. Interhand2. 6m: A dataset and baseline for 3d interacting hand pose estimation from a single rgb image. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 548–564. Springer, 2020.

[17] 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.

[18] Georgy Ponimatkin, Yann Labbé, Bryan Russell, Mathieu Aubry, and Josef Sivic. Focal length and object pose estimation via render and compare. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3825–3834, 2022.

[19] Neng Qian, Jiayi Wang, Franziska Mueller, Florian Bernard, Vladislav Golyanik, and Christian Theobalt. Parametric hand texture model for 3d hand reconstruction and personalization. 2020.

[20] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020.

[21] Jinwei Ren, Jianke Zhu, and Jialiang Zhang. End-to-end weakly-supervised multiple 3d hand mesh reconstruction from single image. *arXiv preprint arXiv:2204.08154*, 2022.

[22] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022.

[23] Michael Seeber, Roi Poranne, Marc Polleyfeys, and Martin R. Oswald. Realistichands: A hybrid model for 3d hand reconstruction. In *2021 International Conference on 3D Vision (3DV)*, pages 22–31, 2021.

[24] Michael Seeber, Roi Poranne, Marc Polleyfeys, and Martin R Oswald. Realistichands: a hybrid model for 3d hand reconstruction. In *2021 International Conference on 3D Vision (3DV)*, pages 22–31. IEEE, 2021.

[25] Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. Transductive semi-supervised deep learning using min-max features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 299–315, 2018.

[26] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1145–1153, 2017.

[27] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[28] Xiao Tang, Tianyu Wang, and Chi-Wing Fu. Towards accurate alignment in real-time 3d hand-mesh reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11698–11707, October 2021.

[29] Linlin Yang, Shicheng Chen, and Angela Yao. Semihand: Semi-supervised hand pose estimation with consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11364–11373, 2021.

[30] 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.

[31] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 3d hand pose tracking and estimation using stereo matching. *arXiv preprint arXiv:1610.07214*, 2016.

[32] 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/CVF International Conference on Computer Vision*, pages 2354–2364, 2019.

[33] Zhedong Zheng and Yi Yang. Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation. *International Journal of Computer Vision*, 129(4):1106–1120, 2021.

[34] 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.

[35] 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/CVF International Conference on Computer Vision*, pages 813–822, 2019.