APSIPA Transactions on Signal and Information Processing, 2024, 13, e403

Original Paper

Estimating 3D Hand Poses and Shapes from Silhouettes

Li-Jen Chang[1], Yu-Cheng Liao[1], Chia-Hui Lin[1], Shys-Fang Yang-Mao[2] and Hwann-Tzong Chen[1*]

[1] *National Tsing Hua University, Taiwan.*
[2] *Electronic and Optoelectronic System Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan.*

ABSTRACT

We present Mask2Hand, a self-trainable method for predicting 3D hand pose and shape from a single 2D binary silhouette. Without additional manual annotations, our method uses differentiable rendering to project 3D estimations onto the 2D silhouette. A tailored loss function, applied between the rendered and input silhouettes, provides a self-guidance mechanism during end-to-end optimization, which constrains global mesh registration and hand pose estimation. Our experiments show that Mask2Hand, using only a binary mask input, achieves accuracy comparable to state-of-the-art methods requiring RGB or depth inputs on both unaligned and aligned datasets.

Keywords: Hand Pose Estimation, Hand Shape Estimation, Differentiable Rendering

1 Introduction

Hundreds of years ago, the historic art form of hand shadow puppetry was already documented in many ancient countries. Since then, making hand

*Corresponding author: htchen@cs.nthu.edu.tw

Received 28 February 2024; revised 07 July 2024; accepted 22 August 2024
ISSN 2048-7703; DOI 10.1561/116.20240013

shadow puppets has been a low-cost household entertainment activity that parents can play with their children. Regarding human perception, it is interesting that humans can recognize the animals and characters portrayed by the hands and fingers from merely their projected shadows. It is also remarkable that people can develop and create different ways of making hand shadow puppets, which may involve the generative capability of imagining some target shape and adjusting the hands and fingers to match the imaginary shape with the shadow. Inspired by the discriminative and generative nature of the visual process of making hand shadow puppets, we aim to explore the possibility of learning to estimate the 3D hand pose and shape from an input hand silhouette through a similar trial-and-error manner.

Our primary goal in this endeavor is to train a renderer-equipped deep network that takes a binary mask of a hand's silhouette as the input and predicts the corresponding 3D hand shape that produces a very similar silhouette. (Figure 1 shows several example results obtained using our model.) To build such a model, we employ MANO [25] as the 3D hand-shape renderer in our deep network. We propose a new network called Mask2Hand, which consists of an encoder, a MANO layer, a differentiable render, and the refinement module. Figure 2 shows an overview of the proposed Mask2Hand network. Our network takes a binary mask as the input and uses the encoder to generate the required parameters for the MANO layer to render the hand mesh. The encoder learns to predict the pose-related principal components of MANO and the 3D global transformation of the hand from only a single input binary mask. Based on the principal components and global transformation, the MANO layer reconstructs the 3D hand mesh and the hand joint positions. We then use a differentiable renderer to generate the 2D projected silhouette from the hand mesh. Finally, the refinement module compares the projected silhouette with the input binary mask to refine the entire network. Three groups of loss functions are tailored for training the aforementioned modules, including pose loss, silhouette loss, and mesh loss. The loss functions and the operations involved in the modules are all differentiable—we are able to train the entire network from end to end. Our code can be found on GitHub by searching Mask2Hand.

As this work aims at addressing a rather new and challenging task of predicting the 3D hand pose from a single 2D binary mask, it is quite encouraging that the proposed Mask2Hand network can achieve comparable prediction accuracy on 3D hand pose and shape estimation as state-of-the-art methods, although we only need a single binary input image while theirs require an RGB image or a depth map. Note that our model can be trained under supervision with manually annotated joint positions or fully self-supervised without human annotations. Furthermore, since our method learns to predict the global transformation of the hand, its evaluation performance under the unaligned setting of hand pose estimation does not degrade as much as other methods do. Such a property is particularly favorable as, in real applications, it is not allowed to align the prediction with the unknown ground truth.

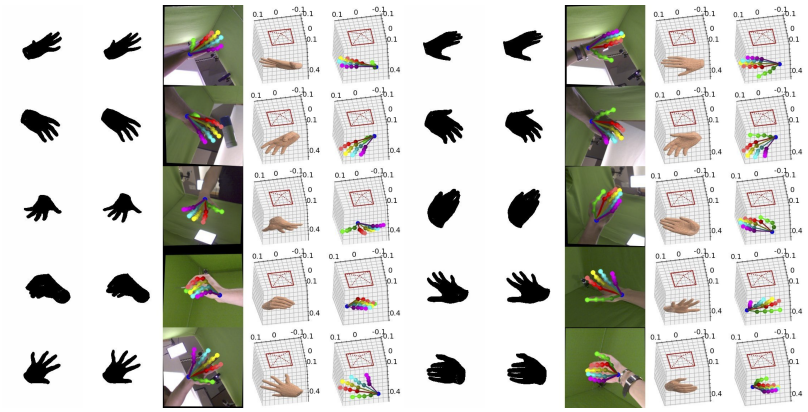

Figure 1: The proposed method, Mask2Hand, learns to solve the challenging task of predicting the 3D hand pose and shape from a 2D binary mask of hand silhouette. Each row in the left or right half of the figure illustrates an example of our method being performed on real data. The first column shows the input of the 2D binary mask. The second column shows the rendered mask using the predicted hand shape. The third column shows the predicted 2D skeleton depicted in the image space. Note that the color images are simply for visualization; our method does not use any RGB data. The last two columns show the predicted mesh and 3D skeleton.

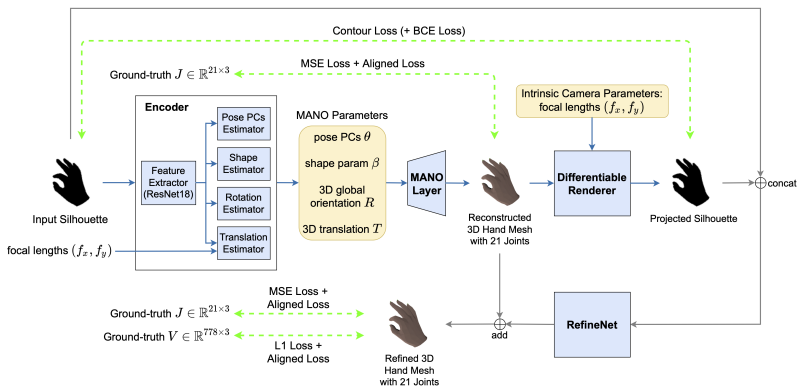

Figure 2: The Mask2Hand network comprises the Encoder, MANO Layer, Differentiable Renderer, and RefineNet. The Encoder takes a binary mask as the input and generates the required parameters for the MANO Layer; the parameters include the shape, the six or forty-five pose-related principal components of MANO, the 3D global orientation, and the translation of the hand. Based on these parameters, the MANO Layer reconstructs the 3D hand mesh (represented by 778 vertices) with the 21 positions of the hand joints. The Differentiable Renderer generates the 2D projected silhouette of the hand mesh, and the RefineNet can compare the projection with the input binary mask to refine the entire model. We design different loss functions for different modules in our model. The green dashed lines indicate what losses are applied to the associated modules. Please refer to the text for the detailed formulations of the loss functions.

In addition to the motivation of hand shadow puppets, we would like to point out several scenarios of real-world applications in which working on binary silhouettes has advantages.

1. **Accessibility with inexpensive low-resolution sensors**: Unlike RGB input, binary images can be obtained with relatively inexpensive sensors, such as low-resolution range and infrared sensors. They are especially useful when the lighting condition is poor and not much color information is available.
2. **Potential for ubiquitous sensing**: A line of research [17, 21, 31] in the hardware domain explores the usage of ambient light as a ubiquitous sensing medium. Light sensors, such as photodiodes or solar cells, measure and track the shadow blockage that the user's hands cause with low cost and ultra-low power consumption, which have the potential to be widely used in IoT devices for shadow-based hand pose estimation systems.
3. **Robustness to adversarial attacks**: Binary input is more robust to adversarial attacks of imperceptible pixel perturbation. From the literature on adversarial machine learning [1], we know that invisible image manipulation can seriously spoil the functionality of DNNs. However, this common type of attack is not directly applicable to the binary setting since small perturbation in pixel values is impossible with binary images. Also, pixel inversion in binary silhouettes can be easily identified and addressed.

2 Related Work

One of the research endeavors inspiring us is *Shadow Theatre* [33], which aims to reproduce the shadow puppet theatre computationally—an ancient shadow-oriented form of performance art—just like how we are motivated by hand shadow puppetry. Won and Lee [33] develop an algorithm to incorporate several features and heuristic rules that are related to shadow generation using human bodies. The proposed method optimizes the poses of human bodies to match the given 2D target shapes. The underlying nonlinear high-dimensional optimization is solved using the proposed heuristic strategies to find a plausible solution with reasonable computing resources. Nevertheless, the computation still requires tens to hundreds of minutes to complete on GPU.

2.1 3D Hand Pose and Shape Estimation

Various methods have been proposed to address the problem of 3D hand pose and shape estimation. Generally, the techniques that are used to solve 3D

human pose estimation can also be applied to 3D hand pose estimation. Based on the input format, we consider the following four categories of methods for 3D hand pose and shape estimation: RGB-based, depth-based, event-camera-based, and silhouette-based methods. Among them, silhouette-based methods are most closely related to this work, where we seek to generate the 3D hand pose and mesh from merely a single binary-valued image of a hand silhouette.

Despite the discrepancy in their input type, many of the methods take 2D joint detection as the basis. Such an approach is promising, and many existing deep-learning-based techniques can be employed. The following is a brief review of recent methods in the four categories.

RGB-based Methods.
Lin *et al*. propose a method called Mesh Transformer (METRO) [19] that explores non-local relationships between vertices and joints to reconstruct a 3D human pose from a single RGB image without using parametric mesh models. This method can also be extended to hand mesh reconstruction. Chen *et al*. [5] propose Camera-space Mesh Recovery (CMR) that generates 3D meshes from a single RGB image by extracting 2D cues, including joint landmarks and silhouettes. Chen *et al*. [6] present a self-supervised 3D hand reconstruction network called S^2HAND that estimates 3D poses from 2D detected keypoints. Liu *et al*. [20] propose a semi-supervised 3D hand-object pose estimation method leveraging spatial-temporal consistency in videos to obtain pseudo-labels for self-training. Li *et al*. [16] propose a hand pose estimation method by using multi-task learning to categorize joints into groups so that different features can be learned to recover the 3D joint locations in a group-wise manner.

Depth-based Methods.
Xiong *et al*. [34] propose the Anchor-to-Joint Regression Network (A2J) that exploits informative anchor points to enhance the generalization ability of 3D pose estimation. Fang *et al*. [9] present a dense-prediction-based 3D hand pose estimation method called JGR-P2O that uses a pixel-to-offset prediction network and a joint graph reasoning module to enable end-to-end training and improve computational efficiency.

Silhouette-based Methods.
Silhouette-based methods have not yet been widely applied to hand/body pose estimation. The task to be solved is more challenging than the settings of RGB-based and depth-based methods. The difficulty comes from the lack of information in the input data, where the shading, color, and depth of the 3D objects are absent in the binary-valued images. Lee *et al*. present the Silhouette-Net [15] method that generates 3D hand poses based on binary-valued silhouettes and can achieve a performance similar to or better than

depth-based methods if multiview silhouettes are provided. However, their method requires the guidance of depth maps during the training stage. In contrast, our method does not require additional depth information during training; it only needs a single-view 2D binary mask of the hand as the input. Despite the challenging setting, estimating the 3D hand pose and mesh from limited binary information in the input is our major goal in this work.

2.2 3D Hand Models

Several hand mesh models have been proposed, such as MANO [25], SphereMeshes [29], and Convex Parts [30]. With the help of these models, hand shape generation and pose estimation can be processed via parametric methods. Typically, a hand model like MANO is designed to parameterize a triangle mesh into pose, shape, and rotation parameters. MANO has been widely used by hand pose estimation methods [3, 2, 7, 8, 12, 23, 27, 32, 35, 36] to train their deep networks; the deep networks learn to predict the hand shape in the target image by regressing to the principal components of the MANO model. As shown in Figure 2, our method also uses the MANO model to construct the target hand shape. By learning to optimize the parameters of MANO meshes, our method achieves the goal of generating the 3D hand mesh from a binary-valued silhouette image.

This paper is an extended version of our previous conference paper, "Mask2Hand: Learning to Predict the 3D Hand Pose and Shape from Shadow" [4]. Mask2Hand is a self-trainable method that accurately predicts 3D hand pose and shape from simple 2D binary silhouettes; it offers significant advantages by eliminating the need for complex RGB or depth data. In this paper, we have substantially revised the conference version with the following enhancements:

- **Detailed Model Description**: We provide a thorough explanation of the model architecture to facilitate implementation and reproducibility.
- **Enhanced Visualizations**: We include additional qualitative results to better illustrate the effectiveness of the method.
- **Robustness Analysis**: A new section explores the impact of noisy input on our model's performance.
- **Limitations Discussion**: We address the limitations of our approach.

3 Our Approach

To predict the hand pose and shape from a binary mask, we propose a two-phase end-to-end trainable network, *Mask2Hand*, that aims to reconstruct 3D

hand meshes in the camera space. Figure 2 illustrates the overall pipeline of our method. In what follows, we detail each pipeline module and describe how we formulate the loss functions.

3.1 Model Architecture

The core modules of our model include an encoder for estimating the MANO parameters, a MANO layer that reconstructs the 3D hand mesh, a differentiable renderer for optimizing the hand pose, and a RefineNet improving the accuracy.

3.1.1 Encoder

Given a hand silhouette image presented as a binary mask, the encoder first uses a ResNet18 [13] backbone to extract features from the input. Then, these features are fed into four different estimators to encode 3D information into the PCA space of MANO [25] and predict the extrinsic camera parameters. More specifically, the output of the encoder consists of the first six or forty-five hand-pose-related principal components $\theta \in \mathbb{R}^6$ or $\mathbb{R}^{45}$ of MANO, shape parameters $\beta \in \mathbb{R}^{10}$, 3D rotation $R \in \mathbb{R}^3$ in axis-angle representation, and 3D translation $T \in \mathbb{R}^3$. The detailed model architecture for our Encoder network is illustrated in Table 1.

Table 1: Architecture for Encoder.

Component	Layer	Output
Feature Extractor	ResNet18	1×512
	Linear(512, 512), ReLU	1×512
Pose PCs Estimator	Linear(512, 256), ReLU	1×256
	Linear(256, #PCs)	$1 \times$ #PCs
Rotation Estimator	Linear(512, 256), ReLU	1×256
	Linear(256, 128), ReLU	1×128
	Linear(128, 3)	1×3
Translation Estimator	Linear(514, 256), ReLU	1×256
	Linear(256, 128), ReLU	1×128
	Linear(128, 3)	1×3
Shape Estimator	Linear(512, 256), ReLU	1×256
	Linear(256, 10)	1×10

3.1.2 MANO Layer

The 3D hand shape is represented by a triangle mesh that contains a vertex set $V \in \mathbb{R}^{778 \times 3}$ and fixed faces F indicating the connection between them. In our

network, the hand mesh and joints $J \in \mathbb{R}^{21\times 3}$ are reconstructed using MANO [25]. Being a low-dimensional parametric hand model, MANO can generate the required hand mesh from parameters including the hand-pose-related principal components θ, shape $\beta \in \mathbb{R}^{10}$, rotation R, and translation T.

3.1.3 Differentiable Renderer

Given intrinsic camera parameters and hand meshes in the camera space, we use the differentiable rendering technique [14] to project 3D estimations into the 2D image space. By applying various losses between the rendered mask of the hand silhouette and the input binary image, we can integrate self-supervision into our end-to-end optimization process for constraining global mesh registration. Here, we use the PyTorch3D [24] implementation of the differentiable renderer. The loss functions will be detailed later in the subsequent sections.

3.1.4 RefineNet

In the previous stage, we only consider several pose-related principal components of MANO for parametrizing hands in a low-dimensional space. The lower degree of freedom may have an expressiveness problem in that the finer details of hand gestures cannot be ideally reconstructed. Therefore, we incorporate a small convolutional neural network called the RefineNet module,[1] into our model to deal with this issue. The network takes the concatenation of the input image and the projected silhouette as input, and it produces residual coordinate values that serve as the point-wise offsets to the previously constructed mesh. The final output of our model is the summation of the originally reconstructed hand mesh and the offset predicted by the RefineNet. The refined joints are then estimated by MANO's joint regressor, given the refined mesh as its input.

3.2 Loss Functions

The complete objective function that we design for training our Mask2Hand model comprises three main loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{pose}} + \mathcal{L}_{\text{silhouette}} + \mathcal{L}_{\text{mesh}}\,, \tag{1}$$

where we have the pose loss $\mathcal{L}_{\text{pose}}$ on the predicted joints, the silhouette loss $\mathcal{L}_{\text{silhouette}}$ on the 2D projected silhouette of the hand, and the mesh loss $\mathcal{L}_{\text{mesh}}$ on the reconstructed 3D mesh.

[1] Our RefineNet module is different from the dense prediction network proposed in [18] despite the same name.

3.2.1 Pose Loss

The pose loss in (1) further consists of two loss functions that use the unaligned and aligned joint positions to evaluate the preliminary and refined joint predictions by the squared L2-norm error:

$$\mathcal{L}_{\text{pose}} = \lambda_{\text{J}}\mathcal{L}_{\text{J}} + \lambda_{\text{alignJ}}\mathcal{L}_{\text{alignJ}}\,, \tag{2}$$

where we have $\mathcal{L}_{\text{J}}$ for the unaligned joint evaluation and $\mathcal{L}_{\text{alignJ}}$ for the aligned joint evaluation, with λ_{J} and λ_{alignJ} as the weight factors. The 3D positions of the K joints are represented as a K by 3 matrix, *i.e.*, $J \in \mathbb{R}^{K\times 3}$. The loss function $\mathcal{L}_{\text{J}}$ of the unaligned joint evaluation is computed as

$$\mathcal{L}_{\text{J}} = \frac{1}{K}\sum_{i=1}^{K}\left\|J_i - \widehat{J}_i\right\|_2^2 + \frac{1}{K}\sum_{i=1}^{K}\left\|J_i - \widehat{J}_i^{\text{ref}}\right\|_2^2\,, \tag{3}$$

where J_i is the ground-truth 3D position of joint i, $\widehat{J}_i$ is the preliminarily reconstructed position of joint i, and $\widehat{J}_i^{\text{ref}}$ is the final prediction after refinement.

For the aligned joint loss, the predicted joints are aligned with the ground truth using orthogonal Procrustes Analysis (PA) [26], and then the L2 loss is applied to the aligned and the ground-truth joints:

$$\begin{gathered}\mathcal{L}_{\text{alignJ}} = \frac{1}{K}\sum_{i=1}^{K}\left\|J_i - \widetilde{J}_i\right\|_2^2 + \frac{1}{K}\sum_{i=1}^{K}\left\|J_i - \widetilde{J}_i^{\text{ref}}\right\|_2^2\,, \\ \text{where } \widetilde{J} = \text{PA}(J,\widehat{J}) \text{ and } \widetilde{J}^{\text{ref}} = \text{PA}(J,\widehat{J}^{\text{ref}})\,.\end{gathered} \tag{4}$$

The procedure of alignment $\text{PA}(J,\widehat{J})$ can be described as follows.

1. Translate the joints so that their mean lies at the origin:
$$J \leftarrow J - \frac{1}{K}\sum_{i=1}^{K} J_i\,, \quad \widehat{J} \leftarrow \widehat{J} - \frac{1}{K}\sum_{i=1}^{K}\widehat{J}_i\,.$$
2. Divide the joints set by its Frobenius norm to remove the uniform scaling component:
$$J \leftarrow J/\|J\|_F\,, \quad \widehat{J} \leftarrow \widehat{J}/\|\widehat{J}\|_F\,.$$
3. Solve the orthogonal Procrustes problem: given matrices J and $\widehat{J}$, find an orthogonal matrix $Q \in \mathbb{R}^{3\times 3}$ that maps J most closely to $\widehat{J}$, *i.e.*, $\min_Q \|JQ - \widehat{J}\|_F$ subject to $Q^\top Q = I_{3\times 3}$. The solution consists of $Q = UV^\top$ and $s = \sum_i \sigma_i$, where $U\Sigma V^\top$ is the singular value decomposition of $J^\top\widehat{J}$ and σ_i's are the singular values.
4. $\widetilde{J} = \left(\widehat{J}Q^\top \times s\right) \times \|J\|_F + \left(\sum_{i=1}^{K} J_i\right)/K$, where the Procrustes output is scaled by the norm and shifted by the mean that we derive from J in the first two steps.

3.2.2 Silhouette Loss

The ideal result for the module of differentiable rendering is that the 2D projected hand silhouette matches the input binary mask exactly. Hence, our silhouette loss is based on the binary cross-entropy (BCE) and the contour loss:

$$\mathcal{L}_{\text{silhouette}} = \lambda_{\text{bce}}\text{BCE}(S, B) + \lambda_{\text{contour}}\text{ContourLoss}(S, B)\,, \tag{5}$$

where S is the rendered silhouette and B is the input binary mask. Our contour loss is a differentiable estimation of Chamfer distance between two contours, calculated as a pixel-wise multiplication between the contour of the rendered silhouette and the pre-computed distance field of the ground-truth silhouette's contour. The detailed steps for computing the Chamfer distance are summarized as follows:

1. Compute the distance transform [10] for the contour ∂B of the ground-truth binary mask B in the dataset to get the distance field $\Psi_{\partial B}$.
2. Apply differentiable binarization to the rendered silhouette image S. In our case, we convert a pixel value $x \in [0, 1]$ in S to an approximately binary value as $\dfrac{1}{1 + e^{-100\,(x-0.5)}} \simeq \{0, 1\}$.
3. Zero out the pixel values that are greater than the threshold 0.5 to deal with the noise near the wrist caused by PyTorch3D differentiable rendering. The result of Steps 2 and 3 is a clean and nearly binarized silhouette $\overline{S}$.
4. Apply Laplacian operator to $\overline{S}$ to get $\Delta\overline{S}$. The contour $\partial\overline{S}$ is then obtained by $\partial\overline{S} = \tanh(\max\{\Delta\overline{S}, 0\})$, which maps the values on the contour to near 1 and all the rest to 0.
5. Do element-wise product between the resulting contour $\partial\overline{S}$ and the distance field $\Psi_{\partial B}$ to retrieve the corresponding distance values and then take the sum to get $\text{ContourLoss}(S, B)$.

All of the aforementioned computations are differentiable. Therefore, we can readily include the silhouette loss in our method and end-to-end train the network.

3.2.3 Mesh Loss

We employ the mesh loss to measure the error in predicting the 3D vertices of the hand mesh. Like the pose loss, the mesh loss is also evaluated on both the unaligned and aligned predictions:

$$\mathcal{L}_{\text{mesh}} = \lambda_{\text{V}}\mathcal{L}_{\text{V}} + \lambda_{\text{alignV}}\mathcal{L}_{\text{alignV}}\,. \tag{6}$$

The unaligned vertex loss $\mathcal{L}_{\mathrm{V}}$ consists of the L1 losses for the predictions from the preliminary and refined meshes:

$$\mathcal{L}_{\mathrm{V}} = \frac{1}{M}\sum_{i=1}^{M}\left\|V_i - \widehat{V}_i\right\|_1 + \frac{1}{M}\sum_{i=1}^{M}\left\|V_i - \widehat{V}_i^{\mathrm{ref}}\right\|_1 , \tag{7}$$

where $V \in \mathbb{R}^{M\times 3}$ is the ground-truth 3D vertices of the hand mesh, $\widehat{V}$ is the preliminarily reconstructed mesh, and $\widehat{V}^{\mathrm{ref}}$ is the final prediction of vertices after refinement.

For the aligned vertex loss $\mathcal{L}_{\mathrm{alignV}}$, the predicted vertices are aligned with the ground truth using orthogonal Procrustes analysis as described in the aligned pose loss (4), and then the L1 loss is applied to the aligned and the ground-truth mesh as

$$\mathcal{L}_{\mathrm{alignV}} = \frac{1}{M}\sum_{i=1}^{M}\left\|V_i - \widetilde{V}_i\right\|_1 + \frac{1}{M}\sum_{i=1}^{M}\left\|V_i - \widetilde{V}_i^{\mathrm{ref}}\right\|_1 , \tag{8}$$
$$\text{where } \widetilde{V} = \mathrm{PA}(V, \widehat{V}) \text{ and } \widetilde{V}^{\mathrm{ref}} = \mathrm{PA}(V, \widehat{V}^{\mathrm{ref}}) .$$

4 Experiments

Since the problem formulation of the task we aim to address differs from those commonly adopted in prior work, we have not found state-of-the-art methods that take exactly the same input setting and could, therefore, be considered for direct comparison with our approach. Note that the work of Lee *et al.* [15] is only posted on arXiv as a preprint without available code. Therefore, we do not include their method in the following comparisons. Moreover, their method requires depth maps during training, which is more restrictive than ours. In our experiments, we first compare our method with the state-of-the-art depth-based model A2J [34] on a synthetic dataset and show that transferring our task for an existing model to solve is probably not as straightforward as it may seem. Further, we use the real data in FreiHAND to compare our method with the three state-of-the-art RGB-based methods: I2L-MeshNet [23], MeshTransformer [19], and CMR (ResNet18) [5]. The results show that our method performs particularly well under the unaligned evaluation setting of 3D hand pose estimation, which is more practical for real applications.

4.1 Datasets

We conduct experiments on the following two datasets.

Synthetic Dataset.
This dataset contains 20,000 training and 2,000 test binary images synthesized using MANO [25, 28]. For each sample in the dataset, its hand-pose principal components $\theta \in \mathbb{R}^6$ are randomly sampled from the uniform distribution between $[-2.0, 2.0)$, and each rotation angle of its 3D global orientation R is randomly sampled from the uniform distribution $[-\pi, \pi)$. We use MANO to generate the corresponding hand shapes from the sampled parameters. The ground-truth 3D coordinates of the 21 hand joints can thus be automatically derived from the MANO-rendered hand shapes.

FreiHAND [37].
It is a common real-world RGB dataset with 32,560 training samples and 3,960 test samples. We binarize the segmentation masks available in the training set and use them as input for our model. However, since its original evaluation set does not provide segmentation masks, we resort to splitting its original training set into a random partition of 26,000 training, 3,280 validation, and 3,280 test images to achieve a reasonably fair evaluation of our method.

For quantitative evaluation, we report the standard metrics used by the FreiHAND dataset on the predictions of 3D joints and meshes:

- **MPJPE**: the *mean per joint position error*, which measures the Euclidean distance (in cm) between the ground-truth joints and the predicted joints.
- **MPVPE**: the *mean per-vertex position error*, which measures the Euclidean distance (in cm) between the ground-truth vertices and the predicted vertices of a mesh.
- **AUC of PCK**: the *area under the curve* of the *percentage of correct keypoints*, which is plotted using 100 equally-spaced thresholds between 0 cm to 5 cm.
- **AUC of PCV**: the *area under the curve* of the percentage of correct vertices.
- **PA-MPJPE** or **PA-MPVPE**: It first applies Procrustes alignment between the ground truth and the prediction and then calculates MPJPE or MPVPE. This metric aims to measure reconstruction error that neglects the effect of global rotation, translation, and scaling.
- **mIoU** and **Dice coefficient**: In addition to the preceding metrics used by the FreiHAND dataset for the evaluation of joint and mesh predictions, we also use the *mean intersection over union* and the *Dice coefficient* to evaluate the similarity between the input binary mask and the mask projected from the predicted hand mesh.

4.2 Comparison with State-of-the-Art Depth-based Methods

To demonstrate that state-of-the-art depth-based models cannot be easily transferred to the task with binary input, we compare our model with A2J [34] on the synthetic dataset. The input image of our model is binarized to $\{0, 255\}$ while that of A2J is binarized to 0 (as the background) and the mean depth of joints to treat the depth-based model fairly. This way, even though the input image is binary-valued, we still provide sufficient statistics of depth information to A2J. As shown in Table 2, for the unaligned setting, our model achieves an MPJPE of 0.87 cm, and A2J reaches an MPJPE of 7.73 cm. The results indicate that A2J fails to learn to localize 3D hand joints accurately when the input is binary, even if the reference mean depth is given. In contrast, our method can recover the positions of the joints in the camera space. In terms of MPJPE after Procrustes alignment, our approach also outperforms A2J by a large margin, as presented in Table 2. Some other depth-based models are not quantitatively compared with ours for the following reasons. V2V-PoseNet [22] takes a 3D voxelized depth map as its input, which is expected to fail for silhouette-based input since the voxelization process cannot work for binary images. Hand PointNet [11] converts a depth map into a 3D point cloud and then uses it as the model's input. Likewise, the conversion procedure cannot function properly for silhouettes, making it inadequate for the task of binary input.

Table 2: Comparison with the depth-based model A2J [34] on the synthetic dataset (PA means Procrustes Aligned).

Method	Input	MPJPE ↓	AUC_J ↑	PA-MPJPE	PA-AUC_J
A2J	Mean depth + 5-view aug.	7.73 cm	0.35	1.93 cm	0.62
Ours	Binary (0, 255) + 5-view aug.	**0.87 cm**	**0.83**	**0.53 cm**	**0.89**

4.3 Comparison with State-of-the-Art RGB-based Methods

For the comparison with state-of-the-art RGB-based methods, we select the leading models from the FreiHAND competition leaderboard at CodaLab,[2] including CMR [5] and MeshTransformer [19]. To make a fair comparison with our method, we use the officially released code to re-train their models on the FreiHAND dataset randomly partitioned by us, as mentioned in Section 4.1, under the settings of both binary and RGB input. The former is for performance observation, while the latter is primarily for convincing proof

[2] https://competitions.codalab.org/competitions/21238

that these models are not trained imperfectly. As presented in Table 3, CMR achieves MPJPE and MPVPE of 4.31 cm with silhouette-based input, while MeshTransformer fails to localize 3D hand joints and mesh in the camera space. Our model reaches MPJPE of 3.56 cm and MPVPE of 3.57 cm, which outperforms the previous two methods by a large margin and is even comparable to the result of CMR with RGB input. The superior performance of our model in the unaligned case indicates that it has a better ability to reconstruct 3D hand mesh in the camera space, which is more practical in real applications. In terms of mIoU and Dice coefficient, our model achieves results similar to those of CMR.

Table 3: FreiHAND (**Not Aligned**). Note that the mIoU and the Dice coefficient are only relevant to the binary input case.

Method	Input	MPJPE (cm)↓	AUC of PCK↑	MPVPE (cm)↓	AUC of PCV↑	mIoU↑	Dice↑
Mesh-Transformer	RGB	68.59	0.00	68.58	0.00	–	–
	Binary	68.59	0.00	68.59	0.00	0.20	0.12
CMR (ResNet18)	RGB	3.49	0.42	3.49	0.42	–	–
	Binary	4.31	0.35	4.31	0.35	**0.90**	**0.90**
Ours (6 PCs)	Binary	**3.56**	0.40	**3.57**	0.40	0.88	0.87
Ours (45 PCs)		**3.56**	**0.41**	**3.57**	**0.41**	0.88	0.88

When the predicted joints and vertices are aligned with the ground truth by Procrustes analysis, our approach reaches PA-MPJPE of 0.68 cm and PA-MPVPE of 0.69 cm, as shown in Table 4. These experimental results are significantly better than CMR's performance under the setting of binary input and are comparable to CMR's results with RGB input. Furthermore, unlike MeshTransformer, which totally fails in the unaligned case, our method performs stably well in both cases. Such a balanced behavior is preferable in our task since localization and detail recovery are vital for projecting a silhouette similar to the input one.

4.4 Qualitative Results

We show qualitative results on the test data. Note that the pictures illustrated in the main paper and this section demonstrate the results of the unaligned case. That is, we do not perform Procrustes alignment for visualization.

In Figure 3 and Figure 4, we show two columns of 15 sets of results side by side. Each set contains five images: the first image presents the input of the 2D binary mask. The second image shows the rendered silhouette using the predicted hand shape. The third image shows the predicted 2D skeleton projected onto the image space. Note that the color images shown here are

Table 4: FreiHAND (**Procrustes Aligned**). The additional entries of PA-MPJPE and PA-MPVPE of I2L-MeshNet with RGB input are copied from the original paper [23] and posted here for ease of reference, which can be considered as a strong baseline.

Method	Input	PA-MPJPE (cm)↓	AUC of PCK↑	PA-MPVPE (cm)↓	AUC of PCV↑
I2L-MeshNet	RGB	0.74	–	0.76	–
Mesh-Transformer	RGB	0.54	0.89	0.58	0.88
	Binary	0.60	0.88	0.64	0.87
CMR (ResNet18)	RGB	0.67	0.87	0.68	0.87
	Binary	0.77	0.85	0.77	0.85
Ours (6 PCs)	Binary	0.73	0.85	0.76	0.85
Ours (45 PCs)		0.68	0.86	0.69	0.86

merely for visualization; our method does not require any RGB data. The last two images depict the predicted mesh and 3D skeleton in the camera space.

4.5 Results on Noisy Silhouettes

As relatively clean masks are used for experiments in the main paper, we aim to explore the robustness of each model under the noisy silhouette setting in this section. To achieve this goal, we first train a Deeplabv3 with ResNet-50 backbone as the hand segmentation model that takes an RGB image of a hand as input and produces its estimated mask. Then, we utilize the estimated masks (*i.e.* noisy binary silhouettes) as inputs to train and evaluate both CMR and our method. Note that the silhouette loss is also calculated using the estimated masks in this experiment. The Deeplabv3 model reaches mIoU of 0.956 on the training data and 0.936 on the test data after being trained for seven epochs with a learning rate of 10^{-4}. Despite the high mIoUs of Deeplabv3 predictions, by manually inspecting the produced silhouettes (see Figure 5 and Figure 6), we discover that finger poses in them are not reconstructed well, which can be considered a strong enough noise injected into the input of target models. As shown in Table 5 and Table 6, our method still outperforms CMR by a large margin in all evaluation metrics. Further, the performance degradation of our model is significantly less than that of CMR in both unaligned and aligned cases, demonstrating that our approach is more robust to noisy input and, thus, more suitable for real-world applications.

4.6 Implementation Details

Our encoder's backbone is based on ResNet18 [13]. PyTorch3D's implementation [24] is employed for our differentiable renderer. We use the Adam optimizer to train our network with a batch size of 32. The initial learning

Figure 3: Each result set contains a 2D binary mask as the input, followed by the predicted mask, 2D skeleton, mesh, and 3D skeleton.

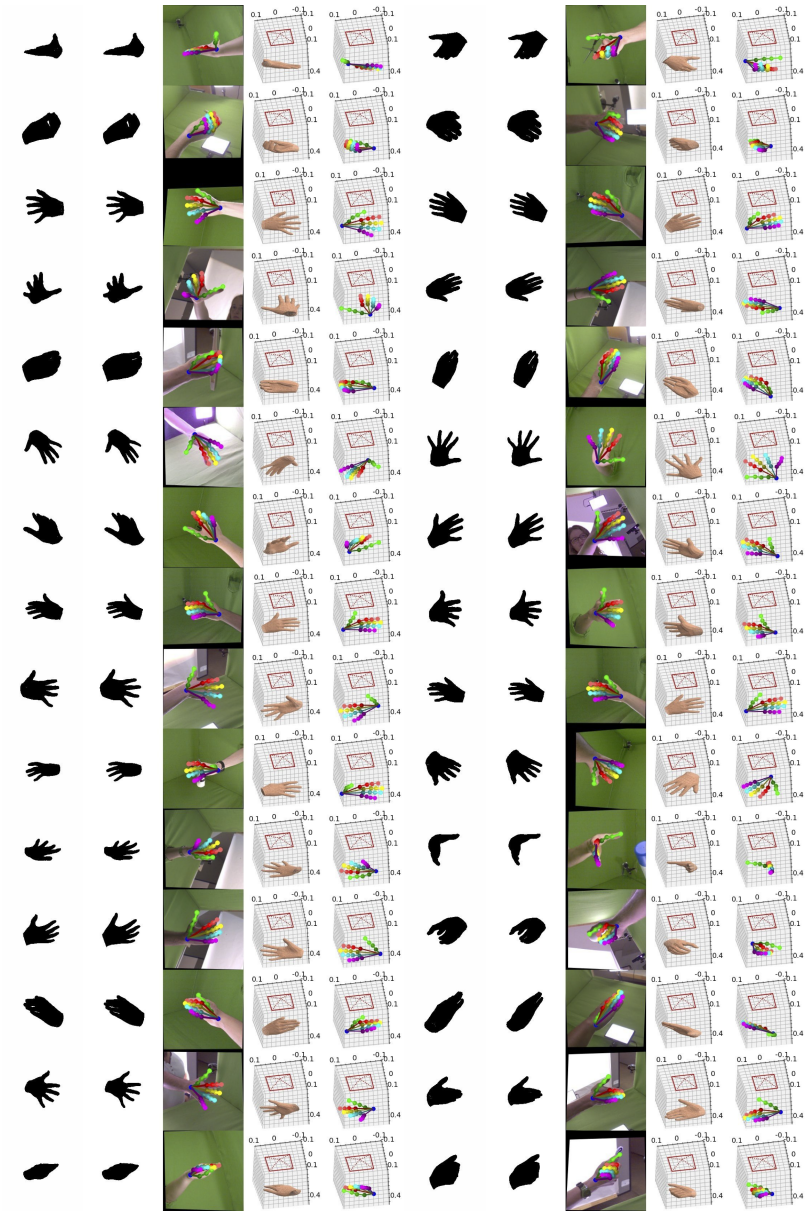

Figure 4: Each result set contains a 2D binary mask as the input, followed by the predicted mask, 2D skeleton, mesh, and 3D skeleton.

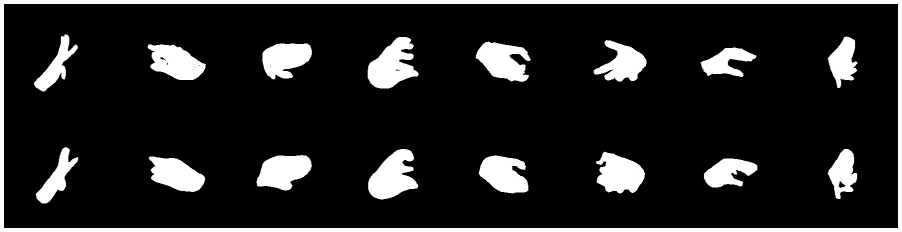

Figure 5: Some bad results of the Deeplabv3 segmentation model. The first row presents the ground-truth masks, and the second row shows the predicted silhouettes. We can observe that finger poses in these examples are not reconstructed well, which can be considered strong noises injected into the inputs of hand pose estimation models.

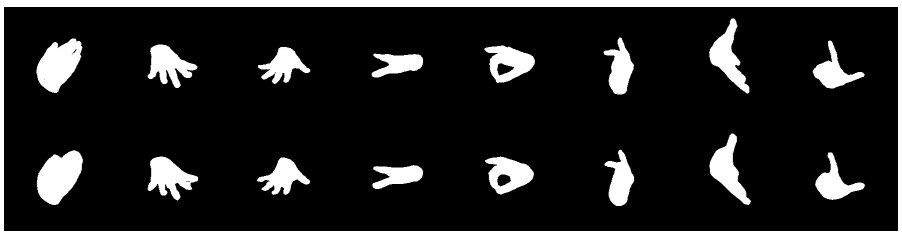

Figure 6: Some good results of the Deeplabv3 segmentation model. The first row presents the ground-truth masks, and the second row shows the predicted silhouettes.

Table 5: Results on noisy silhouettes from FreiHAND (**Not Aligned**).

Method	Input	MPJPE (cm)↓	AUC of PCK↑	MPVPE (cm)↓	AUC of PCV↑
CMR (ResNet18)	Noisy binary silhouette	6.32	0.24	6.33	0.24
Ours (45 PCs)	Noisy binary silhouette	**4.44**	**0.32**	**4.47**	**0.32**

Table 6: Results on noisy silhouettes from FreiHAND (**Procrustes Aligned**).

Method	Input	PA-MPJPE (cm)↓	AUC of PCK↑	PA-MPVPE (cm)↓	AUC of PCV↑
CMR (ResNet18)	Noisy binary silhouette	1.02	0.80	1.04	0.79
Ours (45 PCs)	Noisy binary silhouette	**0.85**	**0.83**	**0.86**	**0.83**

rate is set to 10^{-4}, and we use the ReduceOnPlateau scheduler to adjust the learning rate. The whole network is trained for 150 epochs, and the checkpoint with the lowest validation loss is selected for our final usage. For the FreiHAND dataset, we apply data augmentation of random rotation between $[-\pi, \pi)$ and scaling between $[\ 0.9, 1.1)$ in the 2D image space. For the synthetic dataset, we randomly sample only one of the five views for each training data at every epoch to perform data augmentation. We set $\lambda_{\text{J}} = 2 \times 10^{-3}$, $\lambda_{\text{alignJ}} = 2 \times 10^{-2}$, $\lambda_{\text{contour}} = 10^{-4}$, $\lambda_{\text{V}} = 0.1$, and $\lambda_{\text{alignV}} = 1$ to balance each loss term. λ_{bce} is set to 0 for our final model and 0.5 for the ablation study.

4.7 Ablation Study

We evaluate different configurations of our method to analyze the effectiveness of each component and examine the effects of the loss functions.

4.7.1 Effects of Shape Parameters

In our ablation study, we use six pose PCs and set β to the mean shape, *i.e.*, $\beta = \mathbf{0}$. By comparing the first row of Table 7 and the penultimate row of Tables 3 and 4, we can observe that regressing shape parameters in the first stage of our model helps reduce the burden of refinement and achieve better performance in the aligned case.

Table 7: Ablation Study. ⁕: the aligned loss is only applied to the refined joints and vertices. (A for Aligned Loss, B for BCE Loss, C for Contour Loss, R for Refine Net, M for Mesh Loss.)

A	B	C	R	M	MPJPE (cm)↓	MPVPE (cm)↓	PA-MPJPE (cm)↓	PA-MPVPE (cm)↓	mIoU↑
✓		✓	✓	✓	3.55	3.56	0.78	0.79	0.87
✓		✓	✓		3.67	3.84	0.86	1.18	0.76
✓		✓			3.73	3.73	0.94	0.94	0.83
✓			✓	✓	3.60	3.61	0.80	0.81	0.87
		✓	✓	✓	3.83	3.91	1.36	1.56	0.86
⁕			✓	✓	3.66	3.67	0.79	0.80	0.86
✓	✓	✓	✓	✓	3.56	3.57	0.73	0.76	0.88
✓	✓	✓			3.72	3.73	0.90	0.93	0.83
	✓	✓	✓	✓	3.66	3.67	1.04	1.06	0.88

4.7.2 Effectiveness of Each Component

As shown in Table 7, we explore the evaluation results on the FreiHAND dataset with the exclusion of different components. Our base model consists of

Encoder, MANO Layer, and Differentiable Renderer. After incorporating the RefineNet module into our network with pose loss only, we observe that both MPJPE and PA-MPJPE decrease while MPVPE and PA-MPVPE undesirably increase. This result aligns with our expectation since the pose loss cannot impose strong enough constraints on the deformation of the preliminarily constructed hand mesh. Based on the observation, we further apply the mesh loss to the output of the RefineNet, which helps strengthen the refinement functionality and improve all evaluation metrics significantly.

To reveal the effects of other losses, we exclude them one at a time from our entire network. First, we remove the contour loss between the input binary image and the projected silhouette. The evaluation result shows that the model's performance on both hand pose and shape estimation degrades, especially the unaligned ones, demonstrating the effectiveness of the contour loss for achieving more accurate global mesh registration. Second, we eliminate the aligned losses of both joints and vertices. Our experimental result shows that MPJPE, MPVPE, PA-MPJPE, and PA-MPVPE increase by 0.28, 0.35, 0.58, and 0.77, respectively. Such a significant deterioration in performance indicates that the aligned losses not only play crucial roles in the reconstruction of fine-grained gestures but also help to improve mesh recovery in the camera space. Finally, we remove all losses inserted before RefineNet. The result shown in the sixth row of Table 7 suggests that adding the pose loss and contour loss before the refinement stage can help our model improve the ability to localize the 3D hand mesh in the camera space.

4.7.3 Effects of Adding the BCE Loss

We have also done some experiments to see whether adding the binary cross-entropy loss between the input binary mask and the projected silhouette has any positive effect. As listed in Table 7, directly incorporating the BCE loss into our entire architecture contributes nothing to the final performance. To understand the root cause of this counterintuitive phenomenon, we take away the RefineNet module and the aligned loss one at a time with the presence of BCE loss. The experimental results show that the BCE loss can enhance our model's performance by a large margin when the aligned losses are absent. This implies that the BCE loss does not have much effect when the model learns the shape estimation task to a certain degree guided by the stronger training objectives, *i.e.*, the aligned losses of joints and vertices.

5 Limitations

When there is *apparent* ambiguity in the input silhouette, our method *might* predicts a distinct gesture that gives a very similar projected shadow. For

instance, in the ground-truth image of Figure 7, the middle finger is extended, and the index finger is crooked. Our model predicts inversely, but the resulting silhouette is similar to the ground truth. This issue is quite challenging since the information of gradients in texture is lost in binary images.

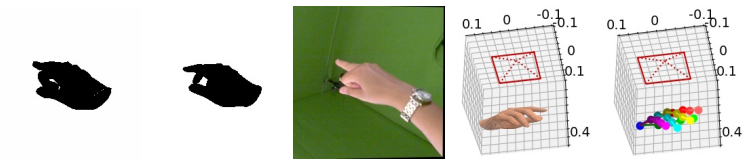

Figure 7: A failure case. The leftmost image is an input 2D binary mask, followed by the predicted silhouette, ground-truth pose, reconstructed mesh, and 3D skeleton.

5.1 Inadequacy for Evaluation on HIM2017 Dataset

We further explain why it is impossible to compare our method against Silhouette-Net on the HIM2017 dataset. HIM2017's depth images include human bodies, and the captured hands are relatively small. The preprocessing step of this dataset requires cropping the hand segments by the given bounding boxes and resizing, which may lead to an unknown transformation between the camera space and the image space. The given camera intrinsics are thus inconsistent and misleading. Such an issue may be one of the reasons that *depth supervision* or *multi-view* is essential for a model (including Silhouette-Net) to succeed in HIM2017. Therefore, we consider HIM2017 inadequate for our task with *single-view* binary information.

6 Conclusion

This paper introduces the Mask2Hand network that learns to predict the 3D hand pose and shape from a 2D binary mask without relying on any RGB or depth information. Despite our task's more challenging input setting compared with prior work on hand pose and shape estimation, we show that the proposed method can achieve comparable performance as state-of-the-art RGB-based and depth-based methods. With the parametric hand model and the differentiable rendering technique, we integrate the self-supervised mechanism into our end-to-end training process without human annotation. We propose several loss functions to model different aspects of 3D hand pose and shape estimation. Since our method explicitly learns to predict the global transformation of the hand, its better performance in the unaligned setting is thus promising for real applications.

Acknowledgments

This work was supported in part by the Industrial Technology Research Institute, Taiwan. We are grateful to the National Center for High-performance Computing for providing computational resources and facilities.

References

[1] N. Akhtar, A. Mian, N. Kardan, and M. Shah, "Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey", *IEEE Access*, 9, 2021, 155161–96.

[2] S. Baek, K. I. Kim, and T. Kim, "Pushing the Envelope for RGB-Based Dense 3D Hand Pose Estimation via Neural Rendering", in *CVPR*, 2019, 1067–76.

[3] A. Boukhayma, R. A. de Bem, and P. H. S. Torr, "3D Hand Shape and Pose From Images in the Wild", in *CVPR*, 2019, 10843–52.

[4] L.-J. Chang, Y.-C. Liao, C.-H. Lin, S.-F. Yang-Mao, and H.-T. Chen, "Mask2Hand: Learning to Predict the 3D Hand Pose and Shape from Shadow", in *APSIPA*, 2023.

[5] X. Chen, Y. Liu, C. Ma, J. Chang, H. Wang, T. Chen, X. Guo, P. Wan, and W. Zheng, "Camera-Space Hand Mesh Recovery via Semantic Aggregation and Adaptive 2D-1D Registration", in *CVPR*, 2021, 13274–83.

[6] Y. Chen, Z. Tu, D. Kang, L. Bao, Y. Zhang, X. Zhe, R. Chen, and J. Yuan, "Model-Based 3D Hand Reconstruction via Self-Supervised Learning", in *CVPR*, 2021, 10451–60.

[7] V. Choutas, G. Pavlakos, T. Bolkart, D. Tzionas, and M. J. Black, "Monocular Expressive Body Regression Through Body-Driven Attention", in *European Conference on Computer Vision (ECCV)*, 2020.

[8] E. Corona, A. Pumarola, G. Alenyà, F. Moreno-Noguer, and G. Rogez, "GanHand: Predicting Human Grasp Affordances in Multi-Object Scenes", in *CVPR*, 2020, 5030–40.

[9] L. Fang, X. Liu, L. Liu, H. Xu, and W. Kang, "JGR-P2O: Joint Graph Reasoning Based Pixel-to-Offset Prediction Network for 3D Hand Pose Estimation from a Single Depth Image", in *European Conference on Computer Vision (ECCV)*, 2020.

[10] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance Transforms of Sampled Functions", *Theory Comput.*, 8(1), 2012, 415–28.

[11] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand PointNet: 3D Hand Pose Estimation Using Point Sets", in *CVPR*, 2018, 8417–26.

[12] Y. Hasson, B. Tekin, F. Bogo, I. Laptev, M. Pollefeys, and C. Schmid, "Leveraging Photometric Consistency Over Time for Sparsely Supervised Hand-Object Reconstruction", in *CVPR*, 2020, 568–77.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", in *CVPR*, 2016, 770–8.

[14] H. Kato, D. Beker, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon, "Differentiable Rendering: A Survey", *CoRR*, abs/2006.12057, 2020.

[15] K. Lee, S. Liu, H. Chen, and K. Ito, "Silhouette-Net: 3D Hand Pose Estimation from Silhouettes", *CoRR*, abs/1912.12436, 2019.

[16] M. Li, Y. Gao, and N. Sang, "Exploiting Learnable Joint Groups for Hand Pose Estimation", in *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021, 1921–9.

[17] T. Li, X. Xiong, Y. Xie, G. Hito, X. Yang, and X. Zhou, "Reconstructing Hand Poses Using Visible Light", *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3), 2017, 71:1–71:20.

[18] G. Lin, F. Liu, A. Milan, C. Shen, and I. D. Reid, "RefineNet: Multi-Path Refinement Networks for Dense Prediction", *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(5), 2020, 1228–42.

[19] K. Lin, L. Wang, and Z. Liu, "End-to-End Human Pose and Mesh Reconstruction with Transformers", in *CVPR*, 2021, 1954–63.

[20] S. Liu, H. Jiang, J. Xu, S. Liu, and X. Wang, "Semi-Supervised 3D Hand-Object Poses Estimation With Interactions in Time", in *CVPR*, 2021, 14687–97.

[21] D. Ma, G. Lan, M. Hassan, W. Hu, M. B. Upama, A. Uddin, and M. Youssef, "SolarGest: Ubiquitous and Battery-free Gesture Recognition using Solar Cells", in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, 12:1–12:15.

[22] 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, 5079–88.

[23] G. Moon and K. M. Lee, "I2L-MeshNet: Image-to-Lixel Prediction Network for Accurate 3D Human Pose and Mesh Estimation from a Single RGB Image", in *European Conference on Computer Vision (ECCV)*, 2020.

[24] N. Ravi, J. Reizenstein, D. Novotný, T. Gordon, W. Lo, J. Johnson, and G. Gkioxari, "Accelerating 3D Deep Learning with PyTorch3D", *CoRR*, abs/2007.08501, 2020.

[25] J. Romero, D. Tzionas, and M. J. Black, "Embodied Hands: Modeling and Capturing Hands and Bodies Together", *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 2017.

[26] P. Schönemann, "A Generalized Solution of the Orthogonal Procrustes Problem", *Psychometrika*, 31(1), 1966, 1–10.

[27] M. Seeber, R. Poranne, M. Polleyfeys, and M. R. Oswald, "RealisticHands: A Hybrid Model for 3D Hand Reconstruction", in *International Conference on 3D Vision (3DV)*, 2021, 22–31.

[28] O. Taheri, N. Ghorbani, M. J. Black, and D. Tzionas, "GRAB: A Dataset of Whole-Body Human Grasping of Objects", in *European Conference on Computer Vision (ECCV)*, 2020.

[29] A. Tkach, M. Pauly, and A. Tagliasacchi, "Sphere-meshes for real-time hand modeling and tracking", *ACM Trans. Graph.*, 35(6), 2016, 222:1–222:11.

[30] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall, "Capturing Hands in Action Using Discriminative Salient Points and Physics Simulation", *Int. J. Comput. Vis.*, 118(2), 2016, 172–93.

[31] A. Varshney, A. Soleiman, L. Mottola, and T. Voigt, "Battery-free Visible Light Sensing", in *Proceedings of the 4th ACM Workshop on Visible Light Communication Systems*, 2017, 3–8.

[32] J. Wang, F. Mueller, F. Bernard, S. Sorli, O. Sotnychenko, N. Qian, M. A. Otaduy, D. Casas, and C. Theobalt, "RGB2Hands: real-time tracking of 3D hand interactions from monocular RGB video", *ACM Trans. Graph.*, 39(6), 2020, 218:1–218:16.

[33] J. Won and J. Lee, "Shadow theatre: discovering human motion from a sequence of silhouettes", *ACM Trans. Graph.*, 35(4), 2016, 147:1–147:12.

[34] F. Xiong, B. Zhang, Y. Xiao, Z. Cao, T. Yu, J. T. Zhou, and J. Yuan, "A2J: Anchor-to-Joint Regression Network for 3D Articulated Pose Estimation From a Single Depth Image", in *IEEE International Conference on Computer Vision (ICCV)*, 2019, 793–802.

[35] X. Zhang, Q. Li, H. Mo, W. Zhang, and W. Zheng, "End-to-End Hand Mesh Recovery From a Monocular RGB Image", in *IEEE International Conference on Computer Vision (ICCV)*, 2019, 2354–64.

[36] Y. Zhou, M. Habermann, W. Xu, I. Habibie, C. Theobalt, and F. Xu, "Monocular Real-Time Hand Shape and Motion Capture Using Multi-Modal Data", in *CVPR*, 2020, 5345–54.

[37] C. Zimmermann, D. Ceylan, J. Yang, B. C. Russell, M. J. Argus, and T. Brox, "FreiHAND: A Dataset for Markerless Capture of Hand Pose and Shape From Single RGB Images", in *IEEE International Conference on Computer Vision (ICCV)*, 2019, 813–22.