Hand and Object Pose Estimation using Self-Supervised Learning

Scientific work to obtain the degree
Master of Science (M.Sc.)

at the Human-centered Assistive Robotics
Technical University of Munich

Submitted by Weiqi Luo
on 26. 08. 2021

First supervisor: Univ.-Prof. Dr.-Ing. Dongheui Lee

Second supervisor: Dr. Hyemin Ahn, M.Sc. Shile Li

TECHNISCHE UNIVERSITÄT MÜNCHEN
Human-centered Assistive Robotics
UNIV.-PROF. DR.-ING. DONGHEUI LEE

March 10, 2020

M A S T E R ' S   T H E S I S
for
Weiqi Luo
Student ID 03697059, Degree EI

Hand and Object Pose Estimation using Self-Supervised Learning

Problem description:

In this project, the main goal is to design systems which can estimate the pose of hand and object. The straightforward method would be training a deep neural network based on a supervised learning framework. However, since obtaining an annotated dataset with an accurate pose is hugely time consuming and difficult, this project would aim to propose a model based on the self- or weakly supervised learning framework. To be specific, the student will develop a self- or weakly supervised method to tackle the hand and object pose estimation problem with help of a differentiable renderer [2] and OpenPose [1]. Without annotated hand and object pose information, the method should regress hand and object pose, and then render an image that matches the input image.

Tasks:

- Literature review on object pose estimation and differentiable rendering.
- Literature review on hand pose estimation and OpenPose-based weakly-supervised learning.
- Self-supervised learning for hand and object pose estimation.
- Evaluation and ablation studies using existing relevant datasets.

Bibliography:

[1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[2] 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.

Supervisor:	Dr. Hyemin Ahn, M.Sc. Shile Li, Prof. Dongheui Lee
Start:	15.07.2020
Intermediate Report:	02.02.2021
Delivery:	26.08.2021

(D. Lee)
Univ.-Professor

Abstract

Hand and object pose estimation is an important topic in robotics and computer vision. State-of-the-art methods typically train a deep neural network on annotated dataset using supervised learning. However, precisely annotating high-dimensional poses in two dimensional image plane is very difficult and time-consuming, especially when there are severe occlusions while the human hand manipulating objects. Some researchers try to eliminate this problem by relying on rendered synthetic dataset. However, the models do not generalize well on the real images due to the domain gap between rendered and real photos.

In this work, we address the issue of lacking annotated dataset for both object and hand pose estimation. We design a self- or weakly supervised learning framework respectively for estimating hand pose and object pose, which directly extract information from input images and use it as an alternative of ground truth. In object pose estimation framework, we utilize a differentiable renderer to render images with estimated poses and train the network by aligning rendered images with input images. For hand pose estimation, we weakly supervise the training process by fitting the MANO hand model to the 2D hand keypoints predicted with pretrained OpenPose hand detector. Through quantitative and qualitative evaluation, we demonstrate that with appropriate settings of objective functions, we can remove the need of pose annotation without losing much accuracy.

Contents

Chapter 1

Introduction

In recent years, hand and object pose estimation has attracted more and more attention thanks to the rapid development of computer vision research. while hand and object pose estimation are essential to interpreting and imitating human actions, it is a key to many applications, such as imitation-based learning of robotic skills, action recognition, human-computer interaction, virtual and augmented reality. Fig. 1.1 demonstrates the usage of hand pose estimation in virtual reality, which enables people to directly manipulate objects in the virtual space without using controllers.

Figure 1.1: Virtual representation of hands in a VR environment powered by Quest augmented reality headset, which is manufactured by Facebook. Hand pose estimation allow people to directly plunge their hands into virtual worlds without the need for a controller.

The objective of object estimation is to estimate the translation and rotation of objects in 3-dimensional space, while the hand pose estimation requires to further predict the posture of hands. Besides, since human hands are different in shape and size and cannot be represented using a uniform model, hand pose estimation is often associated with hand shape estimation. Fig. 1.2 depicts the task of hand and object pose estimation.

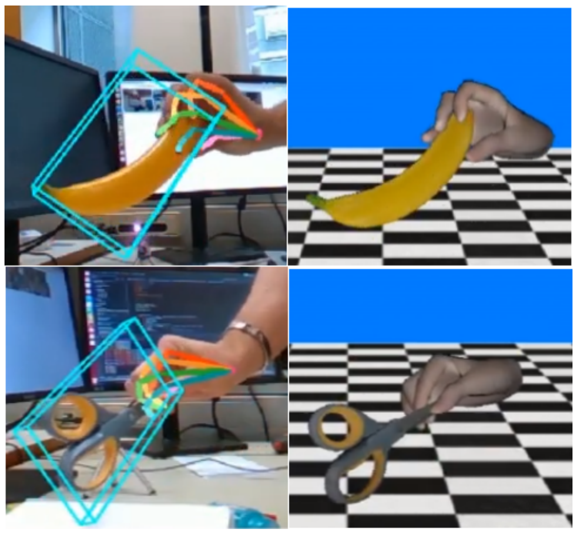


Figure 1.2: Illustration of hand and pose estimation task [9]. The left images show the sensor-captured image, where the 3-dimensional bounding box of the objects are painted as blue boxes, and the skeleton of hands as colored lines. The right images show the reconstruction of the hand and object in the camera coordinate using MANO hand model and CAD model respectively.

Despite the significant progress of deep learning methods made in the computer vision field in recent years [11] [16][34], an accurate estimation of the object and hand pose remains as a challenge. State-of-the-art methods typically rely on training deep neural networks with a supervised learning, which requires a large amount of annotated datasets. Efforts have been made to create datasets with the hand pose labels and object pose labels [46] [8] [39]. Yet annotating geometric labels requires much effort, it is limited due to the time and equipment, and the lacking high-quality annotated dataset limited the performance of the deep neural network.

To overcome the limitation of labeled datasets, a popular method is to use synthetic datasets to train neural networks, where pose labels are automatically generated [10], and then apply them to real images. However, models trained with synthetic datasets do not generalize well to real images. To fill the synthetic-real domain gap, different domain adaption methods are proposed [6] [14]. Nevertheless, the domain gap between synthetic and real data remains difficult to reduce, which greatly limits the performance of methods following this route.

Another direction of solving this problem is to apply self-supervised learning. The idea of self-supervised learning methods is to extract information from the input itself as an alternative form of ground truth, then train the network using the extracted information by fitting and constraining the prediction to be consistent with input data [35] [20].

In this work, we propose two novel self-supervised learning frameworks based on differentiable renderer and OpenPose human keypoints detector to tackle the aforementioned challenges in lacking annotated datasets. The object pose estimation framework is inspired by the recent development of differentiable renderer [24], which allows to reproject the 3D model on the image in a differentiable way, hence the rendering loss can be back-propagated to supervise the pose estimation without the need for a pose-annotated dataset. For the hand pose estimation, since the hand shape and color are different from humans, relying on image alignment is not enough for training the network. Alternatively, we apply a popular human body keypoints detector, OpenPose[3], to firstly extract 21 2-dimensional hand keypoints from RGB images, and then train the network by fitting the hand model to the detected keypoints.

Our contribution can be summarized as follows:

- We design the object pose estimation pipeline for the training network without pose annotation, where the loss function is designed to align the input image and the rendered image.
- We construct the hand pose estimator, which relies on 2D hand keypoints as a weakly supervisory signal.
- We experimentally demonstrate the advantages of the proposed methods both qualitatively and quantitatively through multiple experiments.

Chapter 2

Related Work

In this section, we will review representative works that have been done in related to our topic from four different perspectives: (*i*) related works for object pose estimation, (*ii*) for hand pose estimation, (*iii*) domain adaptation techniques developed for synthetic datasets, and (*iv*) recent advances in self-supervised learning in the field of pose estimation.

Object Pose Estimation.

The work on object pose estimation in the context of deep learning can be mainly categorized into three branches.

One branch of works splits the pose estimation task into two stages. It first establishes 2D-3D correspondences between images and the 3D object model, then obtains the object pose by solving the Perspective-n-Point (PnP) problem. Hu et al. [15] propose a segmentation-driven 6D pose estimation framework, where each visible part of the object in the image makes a prediction of 2D keypoint locations and results in a robust set of 3D-to-2D correspondences. Zakharov et al. [42] establish dense multi-class 2D-3D correspondences maps and refine the initial pose estimation using a deep neural network. Park et al. [26] further improve the performance of keypoints-based methods on the textureless objects by predicting 3D coordinates of each object pixel without textured models, and uses auto-encoder architecture to estimate the 3D coordinates and expected errors per pixel. Despite the good results achieved by these methods, the pose refinement with RANSAC is time-consuming for objects with rich textures.

To avoid heavy computation brought by the keypoints matching, another branch explores the pose embedding and utilizes it for later retrieval. Sundermeyer et al. [35] utilize Augmented Autoencoder to learn an implicit representation of object orientations in a latent space, while Wohlhart et al. [38] train a convolutional neural network by enforcing similarity constraints (CNN) to learn descriptors to capture both the object identity and 3D pose.

While the aforementioned methods rely on a two-stage process, which is not end-to-end trainable, the third branch is to directly regress object pose in an end-to-end manner. Poirson et al. [29] extend SSD object detector to include rough pose estimation in a single shot, without intermediate stages of keypoints matching or object detection, and Kehl et al. [20] further improve the method in accuracy and speed. Since we require the pose estimation to be differentiable such that the self-supervision signal can be back-propagated, we follow the third line of work.

Hand Pose Estimation.

The 3D hand pose estimation is treated as predicting the 3D positions of sparse joints in most of the works. Iqbal et al. [17] propose to learn depth maps and heatmap distributions of hand keypoints with CNN architecture. Li [22] estimate the hand pose from point cloud by merging information from individual points using voting-based scheme. Mueller et al. [25] propose a 3D hand tracking method combining a CNN with a kinematic 3D hand model. Yang et al. [40] leverages other modalities except for pose annotation and color information during the training process and boosts the performance of their hand pose estimation from RGB images.

Recently, predicting 3D full hand mesh has also attracted more and more attention. MANO hand model [30] is one of the most popular hand parameterized models and is used in many works for hand mesh reconstruction. Boukhayma et al. [2] propose to regress the parameters of the MANO model from heatmaps obtained from OpenPose, while Zhang et al. [43] and Baek et al. [1] introduce improvements by iteratively refine the model parameters.

Despite being widely researched, the first branch of work only focuses on the sparse joints position and therefore its application scenarios are limited. To avoid the abovementioned disadvantage, in our work, we aim to estimate both the hand pose and shape. We base our method on the MANO hand model, which has been proven to be an effective way to parametrize the hand pose and shape in 3-dimensional space [2] [43].

Domain adaptation

To overcome the shortage of annotated datasets, some methods render photo-realistic images with perfect labels for training the network and then apply the network on the real sensor-captured images [10]. However, due to the domain gap between the synthetic and the real world, networks that are solely trained with synthetic datasets do not generalize well in the real world. To eliminate this problem, domain randomization techniques are developed to fill the domain gap. Domain randomization adds randomly sampled domain attributes [41] [6] [14] , such as brightness, contrast and background image, to the perfectly rendered images. By training with the augmented dataset, networks become robust to these invariant attributes and can better generalize to the real world. However, the domain randomization is limited to the type of attributes being randomized, the performance will drop if the samples are out of distribution.

Nevertheless, training the model on the synthetic dataset is still an effective way to alleviate the problem of lacking annotated datasets. In our work, we utilize both the real dataset and synthetic dataset, which is generated by rendering photorealistic images with object and hand models. At the same time, to further address the problem of the complicated domain gap between the synthetic and the real world, we focus our work on developing an unsupervised learning technique.

Recent Trends in Self-Supervised Learning

Over the last years, self-supervised learning has achieved promising results in various applications such as depth estimation [7], 3D human pose estimation [4], 3D reconstruction from single RGB images [18], object 6D pose estimation [36], etc. There are also multiple attempts of self-supervised learning made on the pose estimation task. Wang et al. [36] propose to train the network for object 6D pose estimation firstly supervised with synthetic RGB data, then finetuned with self-supervised learning on unannotated real RGB-D data using differentiable neural rendering. They demonstrate a performance outperforming methods relying on synthetic data. Kulon et al. [21] introduced a convolutions-based encoder-decoder structure for monocular 3D hand pose estimation, which is trained with weakly supervisory signal in the form of keypoints.

Encouraged by the progress, we propose to utilize self-supervision in our object and hand pose estimation network to guide the training process of the network. The self supervisory signal consisting of image and geometry alignment information is used to guide the training process, and hence the need for pose annotations is removed.

Chapter 3

Methodology

3.1 Overview

In this chapter, we will first explain the differentiable renderer [24], MANO hand model [30], and 2D hand keypoints estimation of the OpenPose library [3] that we integrated into our framework. Then we will introduce our work in the following sections from the object pose side and hand pose side respectively.

3.2 Background Theory

3.2.1 Differentiable Renderer

To acquire the 2D supervision signal, we revert the 2D-to-3D process by reprojecting the transformed 3D model to a 2D image and back-propagate the difference between rendered and sensor images. While the idea is intuitive, it is not trivial to realize. The reason lies in that the rasterizer of a standard renderer contains two non-differentiable operations. One occurs during drawing pixels, the standard renderer renders a pixel as solid once it is covered by a projected fragment. Another one happens when one pixel is covered by more than one projected fragment. The renderer consider the fragment, which has the smallest depth compared with other overlapping fragments, as visible and use its data to generate this pixel. This process brings one-hot operations into the forward computation of the image rendering, which is not differentiable and prevents the gradient from flowing backward and hence cannot be fit into neural networks.

To address the problem, several works has been done [19] [5] [5] [23] [24]. In the early stage, the focus of the research is to approximate the backward gradient with handcrafted functions, while still employing the standard non-differentiable renderer in the forward rendering process [19][5]. Although the approximation of the backward pass enables the flow of the gradient, the inconsistency between the forward and backward pass harms the performance. Recent advances eliminate the problem by reformulating the rendering process into differentiable functions and calculating the gradient analytically [5][23][24], which have been successfully applied in many applications [36].

In our work, we employ the state-of-the-art differentiable renderer SoftRas [24]. As Fig. 3.1 illustrated, it computes for each mesh triangle a probability map D_j based on the signed distance between pixels and the triangle, and renders the mesh with an aggregating function $\mathcal{A}(\cdot)$ that softly fuses per-triangle color maps based on probabilistic contributions D_j of all mesh triangles as well as the relative depths among triangles. With such formulations, the pixel-level error can be back-propagated as a supervisory signal to train our pose estimation network.

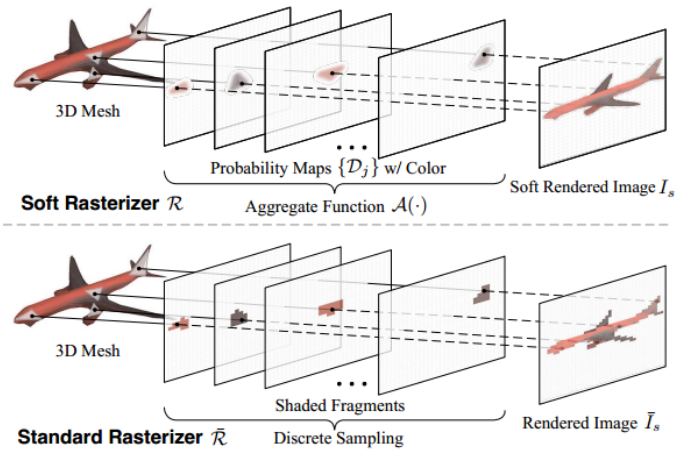

3D Mesh
Probability Maps $\{\mathcal{D}_j\}$ w/ Color
Soft Rendered Image I_s
Soft Rasterizer $\mathcal{R}$
Aggregate Function $\mathcal{A}(\cdot)$
3D Mesh
Shaded Fragments
Rendered Image $\bar{I}_s$
Standard Rasterizer $\bar{\mathcal{R}}$
Discrete Sampling

Figure 3.1: (a) Differentialble rasterizer and (b) standard rasterizer [24].

3.2.2 MANO Hand Model

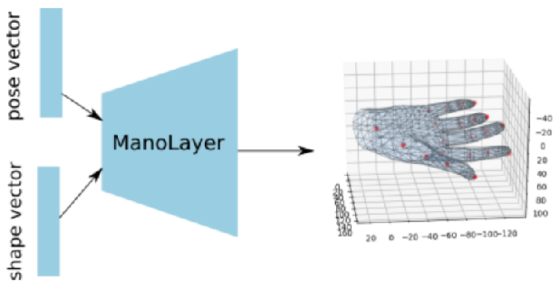


Figure 3.2: Illustration of hand model reconstruction via MANO layer [30].

To fully reconstruct 3D hand mesh, we employ the MANO hand model [30], as depicted in Fig. 3.2. MANO is a differentiable hand model derived from 3D scans of human hands. It is parameterized by 45 pose parameters and 10 shape parameters. The pose parameters capture the rotation angles of 15 hand joints, four points for each finger plus an additional one for the wrist. And the shape parameters control the person-specific deformations of the hand. Fig. 3.4 shows an example of reconstructing hand mesh using the MANO hand model.

Although hands have many degrees of freedom, a large number of combinations are ineffective and can lead to unnatural postures. Therefore, Principal Component Analysis (PCA) [28] is used to reduce the dimension of the hand pose space. According to [32], 3-6 components are enough to account for 80%-90% of the variance in their data. In our work, we use 5 pcas to reconstruct the hand mesh.

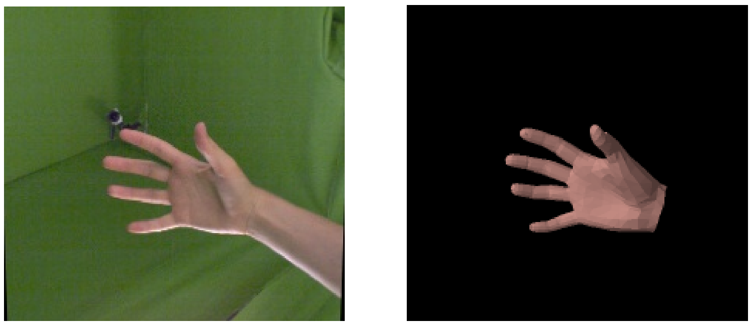

(a) Real image (b) Rendered image with MANO hand model

Figure 3.3: An example of reconstructing hand mesh using the MANO hand model. The left image shows the sensor-captured color image, while the right image shows the corresponding rendered image using MANO hand model.

3.2.3 2D Hand Keypoints Estimation with OpenPose

OpenPose [3] is a multiple-person detection library, which can detect human body, hand, face, and foot keypoints on single images in real-time. In our work, we utilize the hand detector of OpenPose to provide 2D weak supervision for training our hand pose estimator.

OpenPose performs the hand keypoint detection using the architecture called Convolutional Pose Machines (CPMs) [37], which is specially designed for detecting objects that are a composition of different parts such as the human body or human hand, and further improve the performance via multiview bootstrapping [33]. The hand detector takes RGB images as input and estimates the 2D pixel position of 21 hand joints alongside a probability score for each joint.

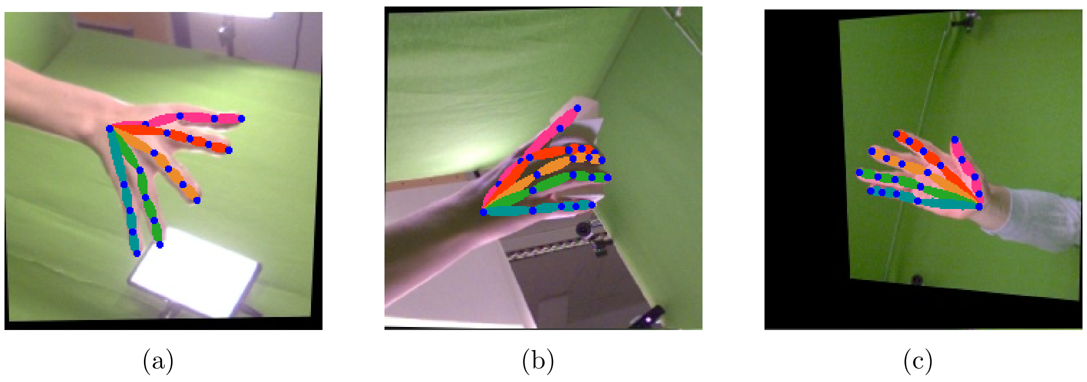


Figure 3.4: Examples of 2D hand keypoints detection using OpenPose. The blue dot in the images represent the detected keypoints, while the lines in different color represent different fingers obtained by connecting hand joints keypoints.

3.3 Object Pose Estimation

The object pose estimation problem is to estimate the 3-dimensional translation and 3-dimensional orientation of the object from given images. To eliminate the problem of lacking annotated datasets for training, we apply differentiable renderers in our project to render photorealistic RGB images and depth images with the estimated object poses, which serves as self-supervision signal and hence removes the need for the object pose annotations.

3.3.1 Network Architecture

The architecture of the object pose estimation network is illustrated in Fig. 3.5. It takes RGB-D images as input and regresses the object pose using an object encoder. The object encoder consist of a ResNet50 [11] pre-trained on ImageNet [31], followed by a 3-layer fully connected network. The differentiable renderer then takes the estimated pose and the object 3D model to render photorealistic images. By utilizing the alignment information between the input and rendered images, we obtain the training loss, which enables self-supervised learning without pose annotation.

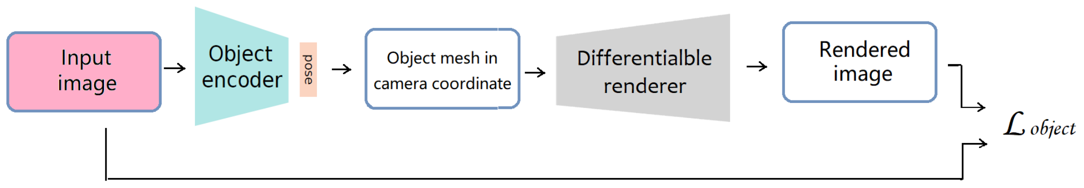


Figure 3.5: Framework of object pose estimation.

3.3.2 Training Loss

In this section, we will introduce the loss function used for training object pose estimation networks. As a comparison, except for the loss function deducted for unsupervised learning, we also formulate the training loss used for supervised learning given the ground truth pose.

Training loss with labeled data

Translation loss. We use L2-norm between the ground truth translation t and estimated translation $\hat{t}$ in 3 dimensional euclidean space to measure their dissimilarity:

$$\mathcal{L}_{trans} = ||t - \hat{t}||_2 \tag{3.1}$$

Quaternion loss. Since the Euler angles suffer from the gimbal lock problem, and the rotation matrix requires additional orthogonality constraints, we choose to use quaternion in our framework to represent the object rotation. While a quaternion q and its opposite $-q$ encode the same rotation, we need to take the sign ambiguity into account when indicating the rotation similarities. A way to eliminate the sign ambiguity is to take the shortest path that connects ground truth quaternion q to estimated quaternion $\hat{q}$ as the distance between them:

$$\mathcal{L}_{quat} = \min(||q - \hat{q}||_2, ||q + \hat{q}||_2) \tag{3.2}$$

In order to improve the training stability, we need to enforce the regressed quaternion stay on the unit sphere. Hence we further include a penalty term in our training loss:

$$\mathcal{L}_{reg} = ||\hat{q} - \mathbf{1}||_2 \tag{3.3}$$

Training loss with labeled data. The training loss when we have labeled dataset is a weighted sum of the translation loss and quaternion loss as well as the regularization term:

$$\mathcal{L}_{labeled} = \alpha_1 \mathcal{L}_{trans} + \alpha_2 \mathcal{L}_{quat} + \alpha_3 \mathcal{L}_{reg}\ , \tag{3.4}$$

where α_1, α_2 and α_3 denote the weight factor for $\mathcal{L}_{trans}$, $\mathcal{L}_{quat}$ and $\mathcal{L}_{reg}$.

Training loss with unlabeled data

LAB loss. Images are originally rendered in the RGB space, where each channel of the image represents the varying level of red, blue, and green brightness. However, RGB format is not optimal for our purpose, since light is the main cause of the domain gap between the rendered and real images, but in RGB format lightness and color are tightly coupled. Therefore, inspired by the work of Self6D [36], we first transfer the images into LAB space. In LAB format, lightness is determined by first channel, while the rest of two channels represent red/green and blue/yellow color respectively. After decoupling light from the color representation, we drop the light channel and evaluate the image similarities only on the Red/Green and Blue/Yellow channel.

Besides light, we also need to eliminate the interference of the background. Here we only take the simple cases into consideration, where the background and object pixels have significantly different depth values. In another word, any pixel with a depth value smaller than a threshold we consider it as object pixel. Assume the mask we obtained from sensor depth image as M^S, we can compute $\mathcal{L}_{ab}$ using the L2 distance function:

$$\mathcal{L}_{ab} = ||I^S_{ab} \odot M^S, I^R_{ab}||_2 \,, \tag{3.5}$$

where I^S_{ab} and I^R_{ab} denotes two color channels of sensor image and rendered image in LAB space.

MS-SSIM (multi-scaled structural similarity). Inspired by the work of Wang et al. [36], we further in clude MS-SSIM loss in our work. SSIM (structural similarity) is a widely used metric for image evaluation [44], which compares two images from the perspective of structural information instead of image color difference. It extracts structural information from images from three aspects: image brightness, contrast and structure similarity, which are estimated using mean, standard deviation and covariance of the image:

$$\mathcal{S}_{ssim}(I_1, I_2) = \frac{(2\mu_1\mu_2 + c_1)(2\sigma_{12} + c_2)}{(\mu_1^2 + \mu_2^2 + c_1)(\sigma_1^2 + \sigma_2^2 + c_2)} \,, \tag{3.6}$$

where $\mu_1, \mu_2, \sigma_1, \sigma_2$ denotes the mean and standard deviation of image I_1 and I_2, σ_{12} denotes the covariance between two images, and c_1, c_2 are small constant values to stable the computation. To further consider the resolution, we utilize MS-SSIM (multi-scaled structural similarity) [44], which is the weighted sum of SSIM calculated on different scale of I_2. Similar to $\mathcal{L}_{ab}$, we first filter out the image background using the object mask.

$$\mathcal{L}_{ms-ssim} = 1 - \mathcal{S}_{ms-ssim}(I^S \odot M^S, I^R) \,. \tag{3.7}$$

Chamfer distance. The depth map can provide geometric information for self-supervised learning. To use this information, we can either directly compare the rendered depth map and the sensor depth map in a 2-dimensional image plane, or first reconstruct the point clouds with the help of the camera intrinsic matrix, and then deduct the loss function in the 3D space. We follow the second routine in our work. The reason is that the geometric information is hard to be fully utilized in the 2D plane, especially at the beginning of the training phase, when the translation error is large and the overlap of the valid part of two images is small. On the contrary, aligning the 3D point cloud can provide much stronger supervision to reduce the translation error, which effectively prevents the model from diverging in the early stages of training. Since it is infeasible to acquire exact 3D-3D correspondences, we refer to Chamfer distance for measuring the distance between two-point clouds. For each point in each cloud, Chamfer distance finds the nearest point in the other point set and computes their Euclidean distance. The distance between two point clouds can be then approximated by averaging the distance of all point pairs.

$$\mathcal{L}_{chamfer} = \frac{1}{|\mathcal{P}^S|} \sum_{p^S \in \mathcal{P}^S} \min_{p^R \in \mathcal{P}^R} ||p^S - p^R||_2 + \frac{1}{|\mathcal{P}^R|} \sum_{p^R \in \mathcal{P}^R} \min_{p^S \in \mathcal{P}^S} ||p^S - p^R||_2 \,, \tag{3.8}$$

where $\mathcal{P}^S$ and $\mathcal{P}^R$ denote the point cloud recovered respectively from the sensor depth image and rendered depth image.

However, in some cases, the estimation error of the object translation in the early training phase is so large, that the object is even out of the camera's view. In this case, neither the rendered depth maps nor the RGB images can provide any effective information. The solution we provide here is to calculate the center of the 3D model point set $\mathcal{M}$ transformed according to the estimated rotation $\hat{R}$ and translation $\hat{t}$, and approximate the point cloud distance using the average distance between the estimated center point and the observed point cloud.

$$\mathcal{L}_{geom} = \begin{cases} \sum_{p^S \in \mathcal{P}^S} \frac{p^S - x_{center}}{|\mathcal{P}^S|}, & \text{if } \mathcal{P}^{\mathcal{R}} = \emptyset \\ \mathcal{L}_{chamfer}, & \text{otherwise} \end{cases} \,, \tag{3.9}$$

where $x_{center} = \frac{1}{|\mathcal{M}|} \sum_{x \in \mathcal{M}} (\hat{R}x + \hat{t})$ denotes the center point of estimated object model.

Training loss with unlabeled data. Intuitively, the more information flows into the network, the faster it learns. Therefore, we use the weighted sum over $\mathcal{L}_{lab}$, $\mathcal{L}_{ms-ssim}$ and $\mathcal{L}_{chamfer}$ as well as the regularization term as the final training loss for self-supervised learning:

$$\mathcal{L}_{object} = \alpha_4 \mathcal{L}_{lab} + \alpha_5 \mathcal{L}_{ms-ssim} + \alpha_6 \mathcal{L}_{geom} + \alpha_7 \mathcal{L}_{reg} \,, \tag{3.10}$$

where α_4, α_5, α_6 and α_7 denote the weighted factor respectively for $\mathcal{L}_{lab}$, $\mathcal{L}_{ms-ssim}$, $\mathcal{L}_{geom}$ and $\mathcal{L}_{reg}$.

3.4 Hand Pose Estimation

Except for the shortage of annotated datasets for training the deep network, which is the same as we encountered in the object pose estimation problem, the hand pose estimation task is much more challenging. The reason lies in two aspects.

On the one hand, unlike the object pose estimation problem, where precise object models are known as prior, there is no uniform hand model available, since the shape and size of hands are different from person to person. Hence, except for the hand pose, we also need to estimate the shape of hands. In our project, we employ the MANO hand model [30] to fully reconstruct the hand mesh, where 10 blend weights are used to model the hand shape.

On the other hand, while the human hand is a multi-rigid body system instead of a single rigid body, we need to specify 15 more rotations for each hand joint except for 3-dimensional global hand location and 3-dimensional global hand orientation. In other words, we need to regress 51 parameters in total to fully determine the hand pose, which is much complex compared with only 6 parameters in the object pose estimation task. However, directly model the joint angles as free variables will lead to impossible hand configurations. Hence, we use Principal Component Analysis (PCA) to exclude physically infeasible hand poses.

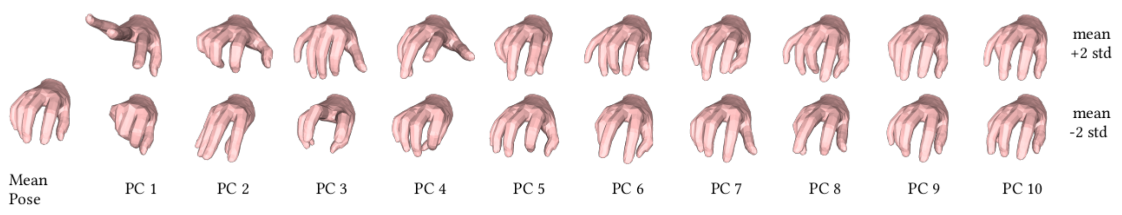


Figure 3.6: Illustration of PCA pose space [30]. The first image shows the rendered MANO hand model with mean pose. The first and second rows of the rest of the columns shows the effect of respectively adding +2 and -2 standard deviations to different principal components.

The high flexibility of hand joints also causes severe self-occlusion, which makes the problem even more complex. Therefore, simply following the same routine as what we did for the object pose estimation is not enough for estimating the hand pose. To provide stronger supervision for hand pose estimation, we utilize a 2D hand keypoints detector from OpenPose. During the training process, the pose and shape parameters are estimated by fitting the MANO hand model to the 21 2D locations of hand keypoints detected using OpenPose from each image.

3.4.1 Network Architecture

The architecture of the hand pose estimation network is illustrated in Fig. 3.7. The backbone of the network is the same as our object pose estimator, which is a ResNet50 [11] pre-trained on ImageNet [31], followed by a fully connected network. The MANO layer then maps the estimated parameters into hand mesh as well as 21 keypoints in the camera coordinate. In parallel with our hand pose autoencoder,

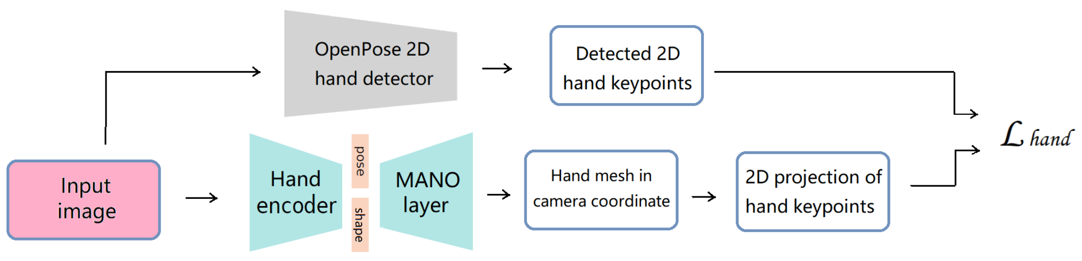

OpenPose 2D hand detector
Detected 2D hand keypoints
$\mathcal{L}_{hand}$
Input image
Hand encoder
pose
shape
MANO layer
Hand mesh in camera coordinate
2D projection of hand keypoints

Figure 3.7: Framework of object pose estimation.

OpenPose hand keypoints detector estimates 21 keypoints locations in pixel coordinates from the input images. The network is then trained by aligning the detected keypoints with the projection of estimated 3D keypoints.

3.4.2 Training Loss

In this section, we will introduce the loss function used for training the hand pose estimation network. Similar to what we did for the object pose estimation, we also formulate the training loss using pose annotation as a comparison.

Training loss with labeled data For training on dataset with MANO parameter annotation, we use L2 norm between the ground truth MANO parameters and estimated MANO parameter for supervised learning. In order to prevent physically infeasible pose estimation, we additionally add regularization term for both shape and pose parameters to penalize deviations from the mean pose.

$$\mathcal{L}_{labeled} = \alpha_8 ||\beta - \hat{\beta}||_2 + \alpha_9 ||\theta - \hat{\theta}||_2 + \alpha_{10} \mathcal{L}_{reg} \tag{3.11}$$

$$\mathcal{L}_{reg} = |||\beta||_{12} + ||\theta||_2 \tag{3.12}$$

where θ and β represent ground truth pose and shape parameters, $\hat{\theta}$ and $\hat{\beta}$ represent estimated pose and shape parameters, and a, b, c, d denote the weighted factor between different terms.

Training loss with unlabeled data Our weakly supervisory signal is provided by 2D landmarks extracted using OpenPose. We define the objective function to align the MANO hand model with the 2D detected hand keypoints, which consists of a join error term, a bone error term, and a regularization term, each of them is multiplied by a weighting factor:

$$\mathcal{L}_{hand} = \alpha_{11} \mathcal{L}_{joint} + \alpha_{12} \mathcal{L}_{bone} + \alpha_{13} \mathcal{L}_{reg} \tag{3.13}$$

The regularization term $\mathcal{L}_{reg}$ is same as equation 3.12. The joints error term $\mathcal{L}_{joint}$ penalize the distance between projection of the 21 keypoints reconstructed with

MANO hand model and the landmarks detected using OpenPose:

$$\mathcal{L}_{joint} = ||\hat{j}_{2D} - j_{2D}||_2 \tag{3.14}$$

where j_{2D} represent the 2D keypoints detected by OpenPose and $\hat{j}_{2D}$ the corresponding estimations, which are the projection of keypoints of reconstructed hand mesh.

Except for the 2D distance between joints, we also employ cosine distance between the bones:

$$\mathcal{L}_{bone} = 1 - \cos(\hat{b_{2d}}, b_{2d}) \tag{3.15}$$

where b_{2d} and $\hat{b}_{2d}$ represent 2D projection of bones, which are obtained by connecting the detected and estimated keypoints in a pre-defined sequence, as shown in Fig. 3.8

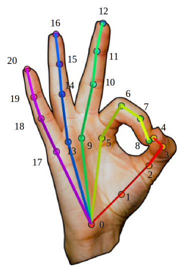


Figure 3.8: Illustration of hand keypoints and bones. Keypoints are represented Bones are shown using colored line segments,

Chapter 4

Evaluation

4.1 Implementation Details

We implement our framework using the deep learning library Pytorch [27]. The network is trained on a single GeForce RTX 2060 with the Adam optimizer. We set the learning rate to 10^{-4} and attenuate it by factor 0.1 after 50th and 80th epoch. For hand pose estimation we train the network with batch size 32, while for object pose estimation we use batch size 24. We resize the input image to 112 by 112 and normalize it with the mean and standard deviation computed from the ImageNet training set. The output of the network is also normalized using Sigmoid function based on the statistics computed from the subset of the training data. The weights of different loss terms are selected empirically as reported in table 4.1:

	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}	α_{13}
value	1	1	0.001	1	0.1	10	0.001	0.1	1	0.001	0.01	1	0.001

table 4.1 weights of loss terms.

4.2 Experiments of Object Pose Estimation

4.2.1 Synthetic LineMOD dataset.

For the object pose estimation task, we generate the synthetic LineMOD dataset using LineMOD dataset [13], which consists of 15 textured 3D object models, as shown in Fig. 4.1. We sample camera rotation randomly and placement within a cylinder of 0.1 m radius and 1 m height. To fill the gap between the synthetic world and the real world, we adopt various data augmentation methods, such as adding random Gaussian noise and Gaussian blending, randomly adjusting brightness, contrast, and hue of the rendered images, and adding random backgrounds.

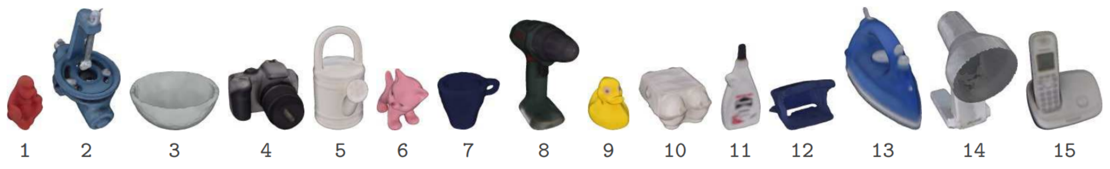


Figure 4.1: LineMOD object models [13].

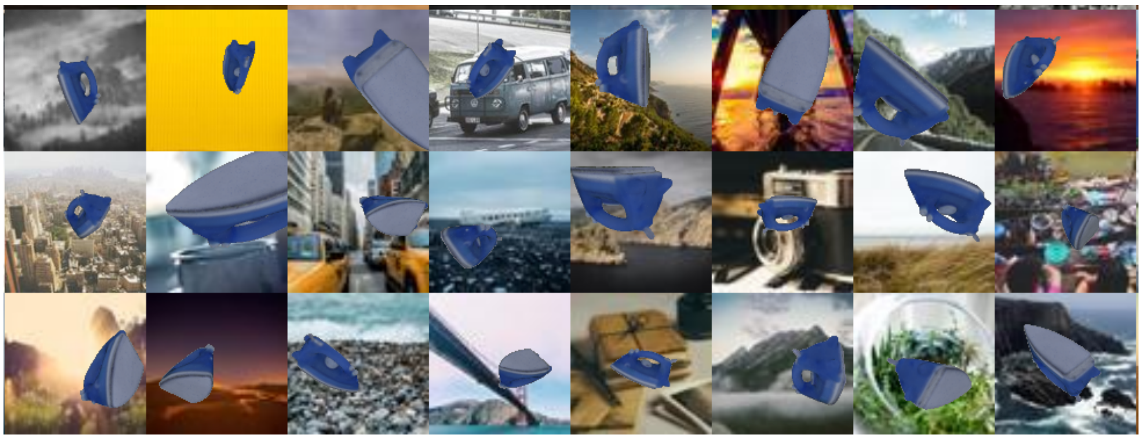


Figure 4.2: Data samples from synthetic LineMOD dataset.

4.2.2 Evaluation Metrics

Mesh error. For object pose estimation, we adopt the average distance metric (mm) designed for evaluating model-based 6D object pose estimation following [12]. Having ground truth rotation R and translation t, as well as estimated rotation $\hat{R}$ and estimated translation $\hat{t}$, the accuracy of object pose estimation is measured using the mean of the pairwise distance between the 3D model point transformed according to the ground truth pose and estimated pose:

$$M_o = \frac{1}{m} \sum_{x \in \mathcal{M}} ||(Rx + t) - (\hat{R}x + \hat{t})||_2, \tag{4.1}$$

where $\mathcal{M}$ denotes the set of 3D model points and m is the number of points.

For symmetrical objects, since the matching between points is ambiguous for some views, the matching score is defined as the average of the closest point distance:

$$M_{so} = \frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} ||(Rx_1 + t) - (\hat{R}x_2 + \hat{t})||_2, \tag{4.2}$$

4.2.3 Experimental Results

To better illustrate the performance of the self-supervised training, we evaluate our framework trained on the synthetic dataset with varying percentages of labeled and unlabeled data. More specifically, we render 10000 images with randomly sampled poses and generate 4 different training datasets by providing annotations to a different percentage of data samples. As illustrated in table 4.2, all the data samples in dataset A are provided with a ground truth pose. From dataset B to dataset C, the percentage of provided annotations decreases. In dataset D, all the data samples are without annotations. For evaluation, we use 400 rendered data samples.

Dataset	labeled samples (%)	unlabeled samples (%)
A	100	0
B	80	20
C	30	70
D	0	100

Table 4.2 Dataset generation.

The percentage of corrected estimated test samples at different epochs is reported in Fig 4.3. The estimation of object pose is considered as correct if the mesh error M_o is smaller than $k_m d$, where k_m is a chosen coefficient and d is the diameter of $\mathcal{M}$. In our experiments, we select k_m as 10%. The x-axis of the figure shows the training timeline in epochs, and the y-axis reports the percentage of corrected estimated samples in the test dataset. Unsurprisingly, the best performance is obtained when the whole training set is labeled. The accuracy is reaching 100% after 100 epochs. Although with unlabeled data the training process is much slower, the accuracy also reaches 90% after 500 epochs. We can also observe that the performance of weakly supervised learning, namely with 80% and 30% labeled data in the training dataset is between self-supervised learning and supervised learning.

To illustrate the source of the error when lacking pose annotations, we plot some evaluation results in Fig. 4.4. As we can see, the number of the failed cases increases as the percentage of pose annotation decreases. Furthermore, we can observe that failure mostly happens when there are other views similar to the ground truth view. The reason lies in that our unsupervised learning method directly learns from the alignment information between rendered and sensor images, hence the network sometimes converges to the local minima in such cases.

The experimental results show that we can utilize a large unlabeled dataset to replace an expensive annotated dataset to train our framework in a self-supervised manner. Although the accuracy suffers a loss compared with supervised learning, considering that in practical cases the label is not precisely labeled, the gap will become smaller.

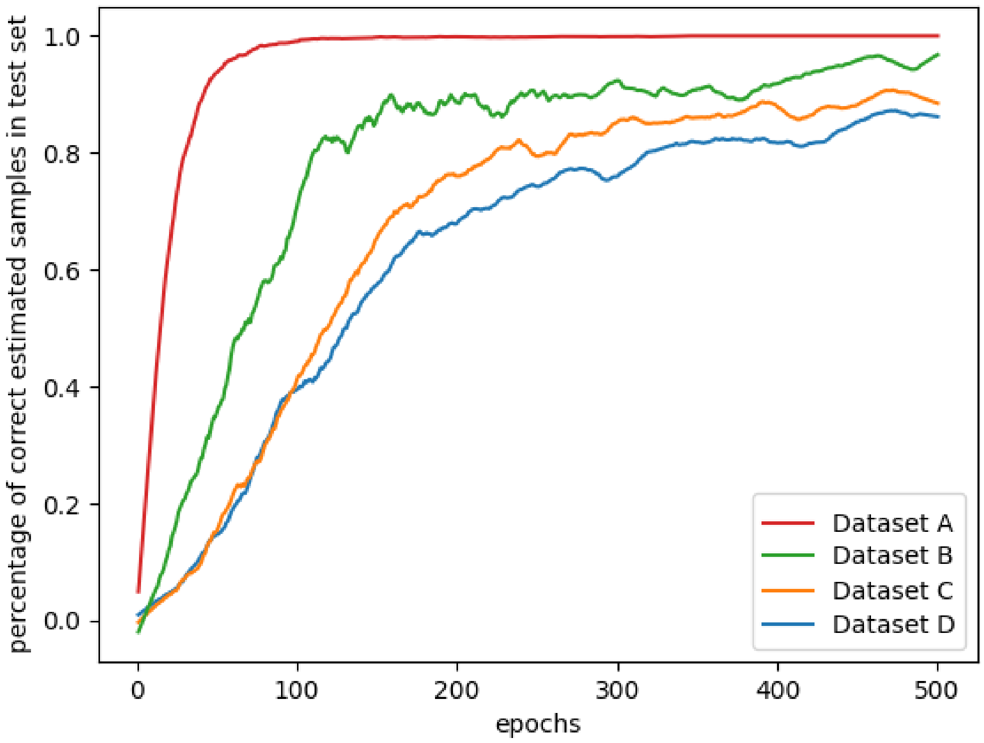


Figure 4.3: Quantitative results of object pose estimation.

4.3 Experiments of Hand Pose Estimation

4.3.1 FreiHAND dataset

FreiHAND[46] is a dataset collected for evaluating and training models for hand pose and shape estimation from RGB images task. It contains 32560 training samples and 3960 evaluation samples. Each training sample contains a color image and a segmentation mask of 224 by 224 pixels, a camera intrinsic matrix, and 3D keypoint annotations for 21 Hand Keypoints, and the evaluation samples provide RGB images and camera intrinsic matrices. While the training samples were recorded with a green screen background and allow for background removal, we further augment the dataset by adding random background during the training process, to improve the generalization ability of our model. Notice that since we are working with self-supervised learning, we did not use any annotation provided by the dataset.

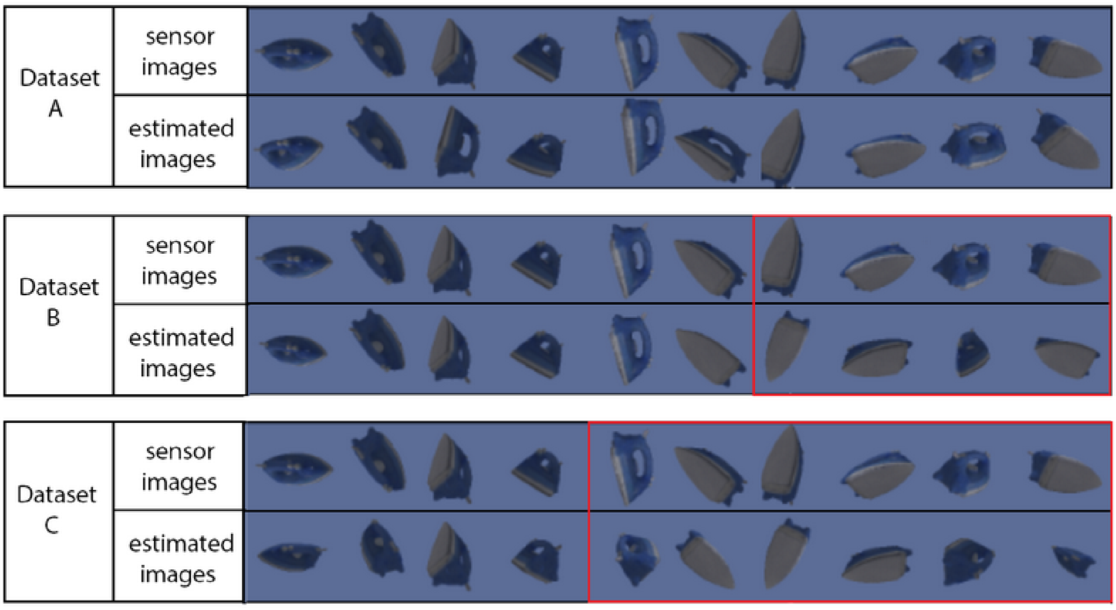


Figure 4.4: Qualitative results of object pose estimation. The images in the red rectangles are failed cases.

4.3.2 Evaluation Metric

We rely on two different evaluation metrics for hand pose and shape estimation to fully capture the performance of our proposed self-supervised learning framework. Notice that since estimating scale from RGB images is an ill-posed problem, during test time, we align our estimation using the reference bone length and camera intrinsics before measuring the pose error.

Pose error. Following [45], We evaluate hand pose estimation using the mean end-point error (mm) over 21 joints, as well as the area under the curve (AUC) of the percentage of correct keypoints (PCK) curve. The x-axis of the PCK curve represents the threshold of the pose error that determines whether a keypoint is corrected estimated, and the y-axis reports the percentage of correctly estimated keypoints. The curve is plotted with 100 evenly spaced thresholds ranging from 0 cm to 5 cm.

$$m_{pose} = \frac{1}{21}\sum_{i=1}^{21} ||j_i - \hat{j}_i||_2, \tag{4.3}$$

where j_i and $\hat{j}_i$ denotes the 3D position ground truth of i-th joint and the corresponding estimation.

Mesh error. While pose error only evaluates the quality of hand pose estimation, to evaluate the hand shape estimation we also report the mean vertices error (mm)

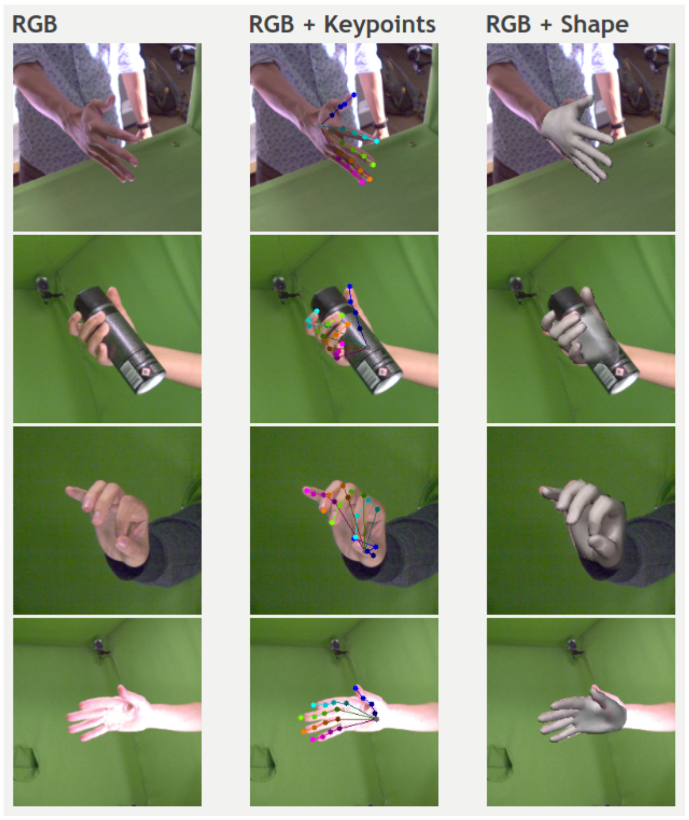


Figure 4.5: Samples in Freihand dataset [46].

over 778 vertices, which are reconstructed using the MANO hand model. Similar to the pose error, we also report the area under the curve (AUC) of the percentage of correct mesh vertices (PCM) curve.

$$m_{mesh} = \frac{1}{778} \sum_{i=1}^{778} ||v_i - \hat{v}_i||_2, \tag{4.4}$$

where v_i and $\hat{v}_i$ denotes the 3D position ground truth of i-th vertice and its corresponding estimation.

4.3.3 Experimental Results

We evaluate our hand pose estimation method on the Freihand dataset [46] and compare it with several supervised-learning-based hand pose estimation methods [46] [2] [10]. The three baselines proposed by Zimmermann et al. [46] are referred to as MANO Mean, MANO Fit, and MANO CNN respectively, which rely on different methods. MANO Fit estimates the MANO parameters by fitting the model to 3D hand keypoints annotations, while MANO Mean follows the same routine but keeps the MANO shape parameters constant as the mean values. Different from MANO Fit and MANO Mean, MANO CNN directly regresses MANO parameters using a convolutional network. Besides the dissimilarity of MANO parameters and joint positions, Hasson et al. [10] also penalize the L2 distance between the ground truth and estimated vertice positions, while Boukhayma et al. [2] penalizes re-projected hand vertices that lie outside of the hand region in a binary mask.

We report both the hand pose error and shape error based on the metrics in 4.3.2. Table 4.3. reports the pose error and pose AUC, while Fig. 4.6 shows the percent of correct keypoints at different values of threshold. We highlight the evaluation results of our method in red color. From the results, we can see that our method outperforms the method developed by Boukh et al.[2] and MANO Mean method by Zimmermann et al. [46] and achieves very close performance compared with the model proposed by Hasson et al. [10] and MANO Fit method by Zimmermann et al. [46]. Although our performance is still worse than MANO CNN proposed by Zimmermann et al. [46], we save a lot of efforts of labelling dataset.

	Ours	MANO Mean[46]	MANO Fit [46]	MANO CNN[46]	Boukh. et al.[2]	Hasson et al.[10]
Pose Error (mm)	1.38	1.71	1.37	1.1	3.5	1.33
Pose AUC	0.727	0.662	0.73	0.783	0.351	0.737

table 4.3 Hand pose estimation performance measured by pose error.

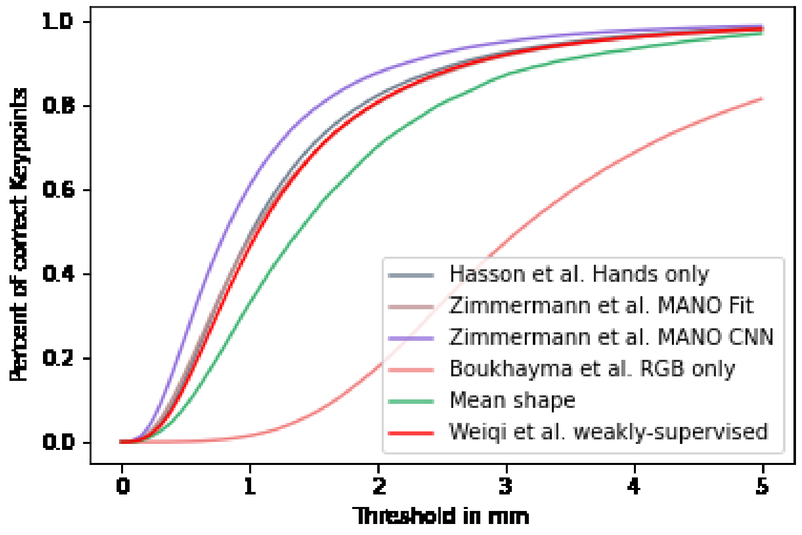


Figure 4.6: Percent of correct keypints curve.

The evaluation results of the shape error and the percent of correct mesh vertices curve are reported in table 4.4. and Fig. 4.7. Similar to the results measured using pose error, our method performs worse than the MANO CNN and better than the MANO Mean method proposed by Zimmermann et al., while is very close to the performance of the other three methods.

	Ours	MANO Mean [46]	MANO Fit [46]	MANO CNN[46]	Boukh. et al.[2]	Hasson et al.[10]
Mesh Error (mm)	1.37	1.64	1.37	1.09	1.32	1.33
Mesh AUC	0.728	0.674	0.729	0.783	0.738	0.736

table 4.4 Hand pose estimation performance measured by shape error.

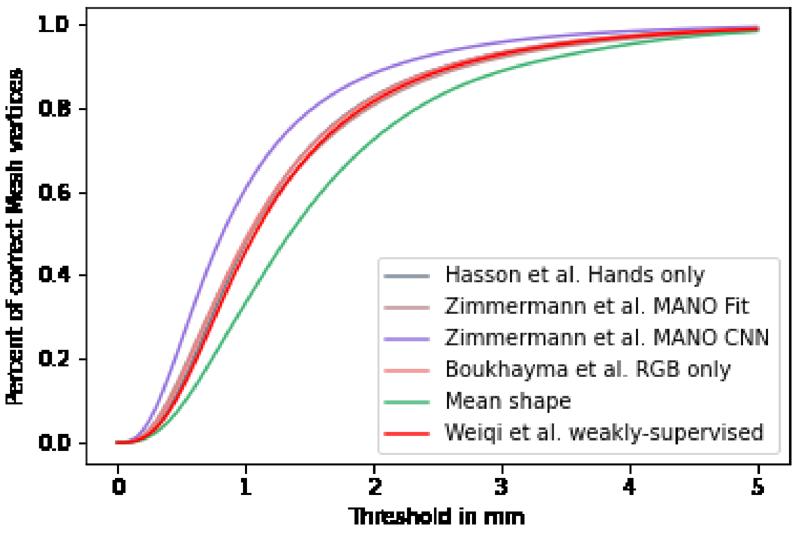
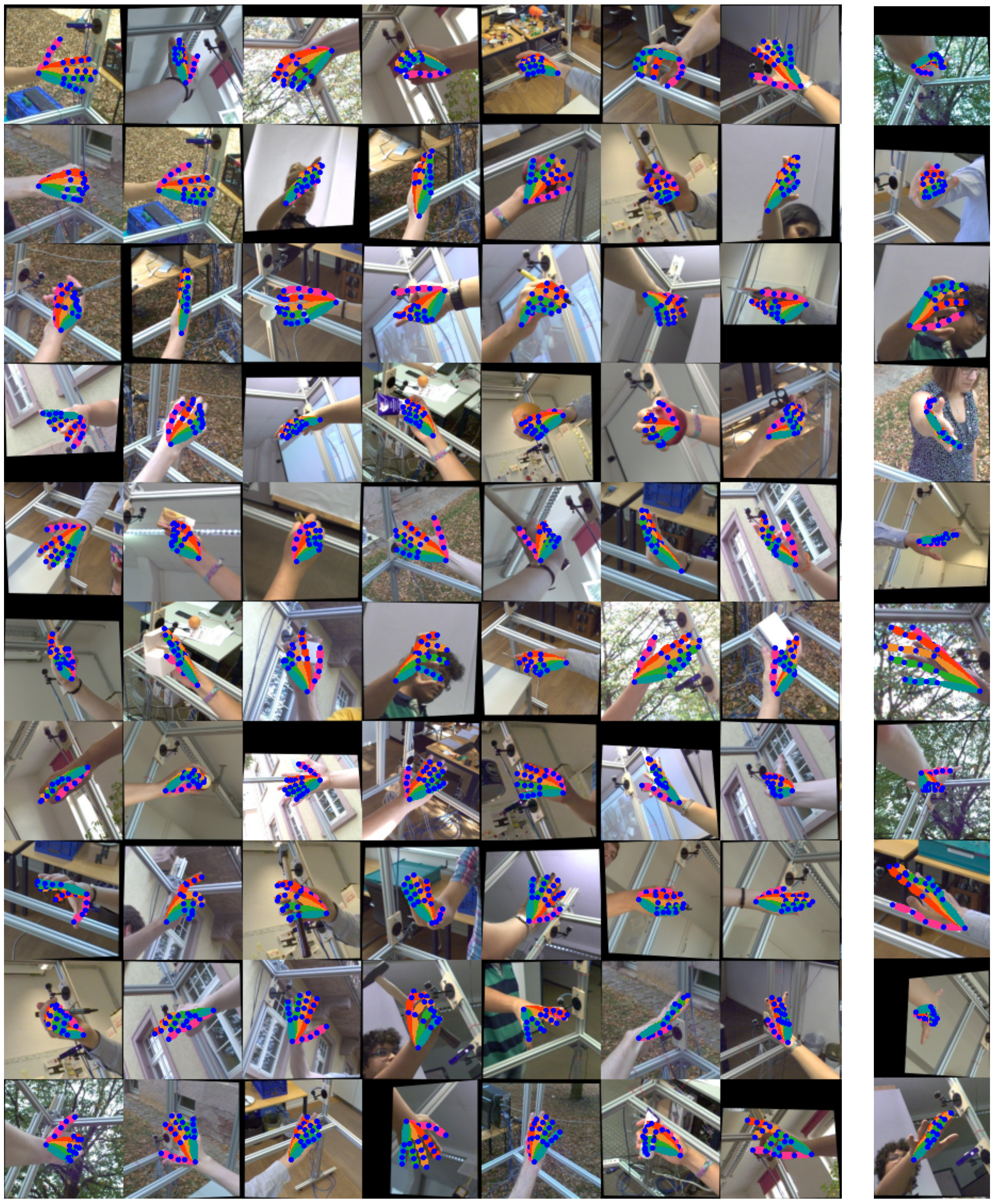


Figure 4.7: Percent of correct mesh vertices curve.

The results have shown that our method has achieved very close performance comparing with other methods which require pose annotations. From the results, we can conclude that our method can remove the need for hand pose labels and save much manpower for annotating pictures while maintaining good performance.

In Fig. 4.8, we plotted some qualitative pose estimation results. The left columns plot the success cases. It can be observed that our pose estimator is able to cope with clustered background and self-occlusion as well as occlusion between hands and objects. However, the pose estimator sometimes also fails. The right columns plot some of the failed cases. We can observe that the network sometimes fails to recover the hand scale or mess up with the hand posture when severe occlusion presents.



(a) Success (b) Failure

Figure 4.8: Qualitative results of hand pose estimation. The hand posture is painted in colored lines.

Chapter 5

Conclusion

Object and hand pose estimation is a widely researched topic, which has made significant progress in recent years. However, the lack of annotated dataset remains an unsolved issue in the hand and object pose estimation and severely harms the performance of deep-learning-based methods.

In this work, we propose two end-to-end frameworks respectively for estimating hand pose and object pose that is able to be trained in a self-supervised manner. In the object pose estimation framework, we utilize the advanced differentiable renderer for photo-realistic rendering, such that we are able to enforce visual and geometrical constraints between the rendered image and the sensor-captured image for training the model. In the hand pose estimation framework, we utilize a hand keypoints detector from OpenPose, a human body 2D pose estimation library to provide 2D locations of hand keypoints as weak supervisory signals. By fitting the reconstructed hand mesh to the 2D keypoints, we are able to estimate the parameters for MANO hand model. We demonstrate both quantitatively and qualitatively that with self-supervisory signals directly extracted from input images, we are able to remove the need for pose annotation without much accuracy loss.

Along this direction, an interesting future work would be to combine hand pose estimation and object pose estimation into a single end-to-end framework allowing for self-supervised learning. Although the self-supervision can eliminate the problem of lacking annotated datasets of pose estimation task, yet estimating the hand-object pose jointly is a more challenging task on account of the significant occlusions of both hand and object. Except for bringing in more challenges, the interaction between hands and objects also provides extra information for joint hand-object pose estimation, since manipulation may limit the space for effective hand-shaped object configuration due to physical contact conditions.

In Fig. 5.1, we illustrate one possible architecture for jointly estimating hand and object pose. The structure is a simple stacking of the hand pose estimator and object pose estimator proposed in our work, as illustrated in Fig. 3.5 and Fig. 3.7 respectively. The simple concatenation of two sub-modules allows us to reuse our model trained for the hand and object pose estimation in our work. Furthermore, combining hand pose estimator and object pose estimator into one piece allows the computation of interaction between hands and objects. By interpreting the interaction between hands and objects as a loss term penalizing penetration and contact between hands and objects, as marked in Fig. 5.1 in red color, we can obtain an extra self-supervisory signal. By jointly training the network for hand and object pose estimation, we can expect a better performance of both hand and object pose estimation.

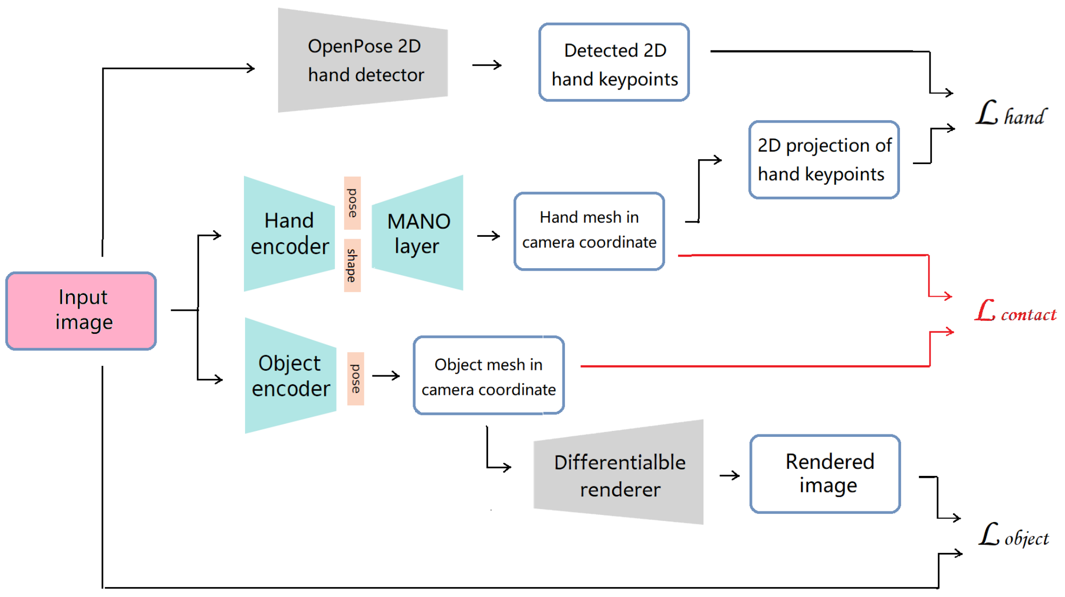


Figure 5.1: Framework of joint hand-object pose estimation.

List of Figures

Acronyms and Notations

HRC	Human-Robot Collaboration
HRI	Human-Robot Interaction
HRT	Human-Robot Team

Bibliography

[1] Baek, S., Kim, K. I., and Kim, T.-K. Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 1067–1076.

[2] Boukhayma, A., Bem, R. d., and Torr, P. H. 3d hand shape and pose from images in the wild. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 10843–10852.

[3] Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. Realtime multi-person 2d pose estimation using part affinity fields. 7291–7299.

[4] Chen, C.-H., Tyagi, A., Agrawal, A., Drover, D., Stojanov, S., and Rehg, J. M. Unsupervised 3d pose estimation with geometric self-supervision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 5714–5724.

[5] Chen, W., Ling, H., Gao, J., Smith, E., Lehtinen, J., Jacobson, A., and Fidler, S. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in Neural Information Processing Systems* (2019), 9609–9619.

[6] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The journal of machine learning research 17*, 1 (2016), 2096–2030.

[7] Godard, C., Mac Aodha, O., and Brostow, G. J. Unsupervised monocular depth estimation with left-right consistency. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), 270–279.

[8] Hampali, S., Oberweger, M., Rad, M., and Lepetit, V. Ho-3d: A multi-user, multi-object dataset for joint 3d hand-object pose estimation. *arXiv preprint arXiv:1907.01481* (2019).

[9] Hampali, S., Rad, M., Oberweger, M., and Lepetit, V. Honnotate: A method for 3d annotation of hand and object poses. 3196–3206.

[10] Hasson, Y., Varol, G., Tzionas, D., Kalevatykh, I., Black, M. J., Laptev, I., and Schmid, C. Learning joint reconstruction of hands and manipulated objects. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 11807–11816.

[11] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. 770–778.

[12] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. 548–562.

[13] Hodan, Tomas, F. M. E. B. W. K. A. G. D. K. B. D. e. a. Bop: Benchmark for 6d object pose estimation. *In Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 47–57.

[14] Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T. Cycada: Cycle-consistent adversarial domain adaptation. 1989–1998.

[15] Hu, Y., Hugonot, J., Fua, P., and Salzmann, M. Segmentation-driven 6d object pose estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 3385–3394.

[16] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. 4700–4708.

[17] Iqbal, U., Molchanov, P., Gall, T. B. J., and Kautz, J. Hand pose estimation via latent 2.5 d heatmap regression. 118–134.

[18] Kanazawa, A., Tulsiani, S., Efros, A. A., and Malik, J. Learning category-specific mesh reconstruction from image collections. *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 371–386.

[19] Kato, H., Ushiku, Y., and Harada, T. Neural 3d mesh renderer. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 3907–3916.

[20] Kehl, W., Manhardt, F., Tombari, F., Ilic, S., and Navab, N. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. 1521–1529.

[21] Kulon, D., Guler, R. A., Kokkinos, I., Bronstein, M. M., and Zafeiriou, S. Weakly-supervised mesh-convolutional hand reconstruction in the wild. 4990–5000.

[22] Li, S., and Lee, D. Point-to-pose voting based hand pose estimation using residual permutation equivariant layer. 11927–11936.

[23] Liu, S., Chen, W., Li, T., and Li, H. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. *arXiv preprint arXiv:1901.05567* (2019).

[24] Liu, S., Li, T., Chen, W., and Li, H. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *Proceedings of the IEEE International Conference on Computer Vision* (2019), 7708–7717.

[25] Mueller, F., Bernard, F., Sotnychenko, O., Mehta, D., Sridhar, S., Casas, D., and Theobalt, C. Ganerated hands for real-time 3d hand tracking from monocular rgb. 49–59.

[26] Park, K., Patten, T., and Vincze, M. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. 7668–7677.

[27] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems 32* (2019), 8026–8037.

[28] Pearson, K. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science 2*, 11 (1901), 559–572.

[29] Poirson, P., Ammirato, P., Fu, C.-Y., Liu, W., Kosecka, J., and Berg, A. C. Fast single shot detection and pose estimation. 676–684.

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

[31] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision 115*, 3 (2015), 211–252.

[32] Schröder, M., Maycock, J., Ritter, H., and Botsch, M. Real-time hand tracking using synergistic inverse kinematics. 5447–5454.

[33] Simon, T., Joo, H., Matthews, I., and Sheikh, Y. Hand keypoint detection in single images using multiview bootstrapping. 1145–1153.

[34] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[35] Sundermeyer, M., Marton, Z.-C., Durner, M., Brucker, M., and Triebel, R. Implicit 3d orientation learning for 6d object detection from rgb images. *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 699–715.

[36] Wang, G., Manhardt, F., Shao, J., Ji, X., Navab, N., and Tombari, F. Self6d: Self-supervised monocular 6d object pose estimation. *arXiv preprint arXiv:2004.06468* (2020).

[37] Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. Convolutional pose machines. 4724–4732.

[38] Wohlhart, P., and Lepetit, V. Learning descriptors for object recognition and 3d pose estimation. 3109–3118.

[39] Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199* (2017).

[40] Yang, L., Li, S., Lee, D., and Yao, A. Aligning latent spaces for 3d hand pose estimation. 2335–2343.

[41] Zakharov, S., Kehl, W., and Ilic, S. Deceptionnet: Network-driven domain randomization. 532–541.

[42] Zakharov, S., Shugurov, I., and Ilic, S. Dpod: 6d pose object detector and refiner. 1941–1950.

[43] Zhang, X., Li, Q., Mo, H., Zhang, W., and Zheng, W. End-to-end hand mesh recovery from a monocular rgb image. *Proceedings of the IEEE International Conference on Computer Vision* (2019), 2354–2364.

[44] Zhao, H., Gallo, O., Frosio, I., and Kautz, J. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging 3*, 1 (2016), 47–57.

[45] Zimmermann, C., and Brox, T. Learning to estimate 3d hand pose from single rgb images. *Proceedings of the IEEE international conference on computer vision* (2017), 4903–4911.

[46] Zimmermann, C., Ceylan, D., Yang, J., Russell, B., Argus, M., and Brox, T. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. *Proceedings of the IEEE International Conference on Computer Vision* (2019), 813–822.

License