Hand Pose Estimation for Hand-object Interaction Cases using Augmented Autoencoder

handed in
MASTER'S THESIS

Haojie Wang

born on the 13.07.1994
living in:
Felsennelkenanger 19
80937 Munich
Tel.: 015253261116

Human-centered Assistive Robotics
Technical University of Munich

Univ.-Prof. Dr.-Ing. Dongheui Lee

Supervisor:	M.Sc. Shile Li, Prof. Dr. Dongheui Lee
Start:	01.03.2019
Intermediate Report:	27.05.2019
Delivery:	14.10.2019

Abstract

Hand pose estimation with objects is challenging due to object occlusion and the lack of large annotated datasets. To tackle these issues, we propose an Augmented Autoencoder based deep learning method using augmented clean hand data. Our method takes 3D point cloud of a hand with an augmented object as input and encodes the input to latent representation of the hand. From the latent representation, our method decodes 3D hand pose and we propose to use an auxiliary point cloud decoder to assist the formation of the latent space. Through quantitative and qualitative evaluation on both synthetic dataset and real captured data containing objects, we demonstrate state-of-the-art performance for hand pose estimation with objects.

Contents

1 Introduction 5

2 Related Work 9

3 Methodology 13
- 3.1 Overview 13
- 3.2 Data Processing 14
  - 3.2.1 View Normalization 14
  - 3.2.2 Data Augmentation 15
- 3.3 Implementation 18
  - 3.3.1 Augmented Autoencoder 19
  - 3.3.2 Variational Autoencoder 20
  - 3.3.3 Residual Permutation Equivariant Layer based Encoder 21
  - 3.3.4 Decoders for Points Reconstruction 22
  - 3.3.5 Fully-connected Layer based Decoder for Pose Estimation 25
  - 3.3.6 Training Loss 25

4 Experiment and Evaluation 29
- 4.1 Training Details 29
- 4.2 Datasets 29
- 4.3 Evaluation Metrics 31
- 4.4 Segmentation using Importance Term 33
- 4.5 Comparison of Reconstruction Decoders 34
- 4.6 Comparison to state-of-the-art Method 35
- 4.7 Ablation Studies 36
- 4.8 Qualitative Results 38
- 4.9 Runtime 38

5 Future Work 45
- 5.1 Disadvantages of Our Method 45
- 5.2 Optimization Methods 46

6 Conclusion 49

A DVD 51

List of Figures 53

Bibliography 57

Chapter 1

Introduction

Hand pose estimation plays an important role in many human-robot interaction tasks, such as teleoperation, virtual/augmented reality and robot imitation learning [ZRG+07][JNC+15][PCBC13][ACVB09]. These applications require real-time and accurate hand pose estimation in 3D space. Recently, deep learning based methods have made significant progress in this area, which can be categorized to depth-based approaches [LL19][GCWY18][GLYT16][GLYT17][WPVGY17][WPVGY18][OWL15] and RGB-based approaches [ZB17][YY19][BBT19][POA18][MBS+18][lLLY19]. Despite the success of these methods, they rarely concern the hand-object interaction cases. These methods typically fail in manipulation tasks because of the occlusions caused by the grasped object. (Fig. 1.1)

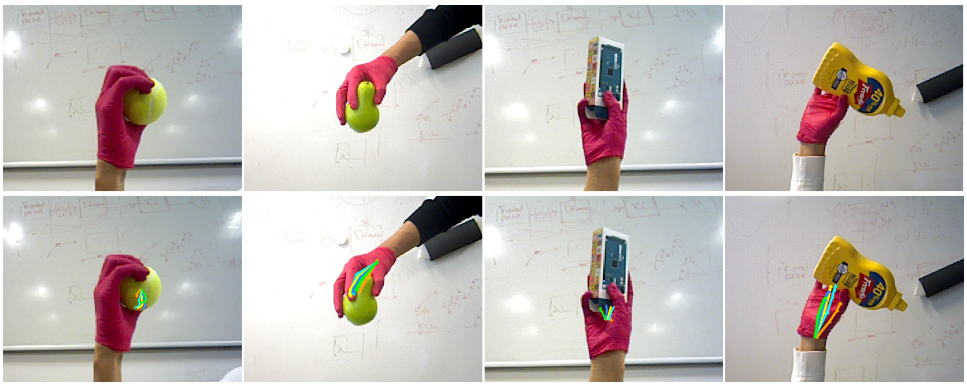


Figure 1.1: A previous hand pose estimation work [LL19] fails in hand-object interaction tasks.

Recently, several works start to take object occlusion problems for hand pose estimation task into consideration. The majority are tracking based approaches [SMZ+16][KA13][TBS+16][HSKMVG09][OKA11b]. The robust performance of these methods relies on tracking algorithms to exploit the temporal constraints between consecutive frames in input sequence. However, a good initialization is required for

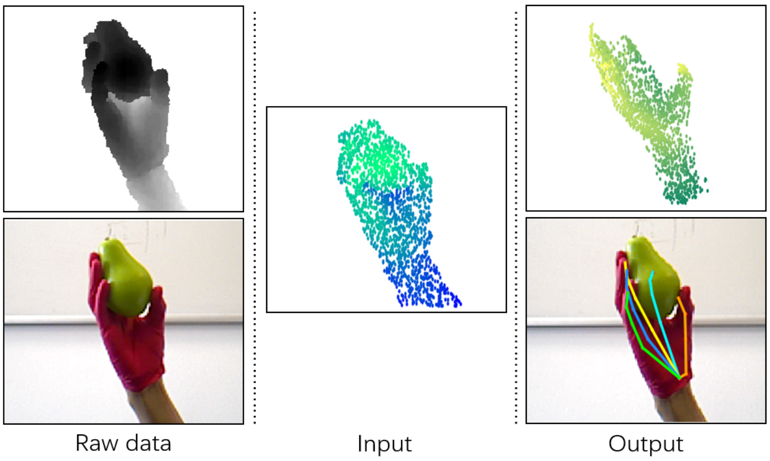


Figure 1.2: The raw data are captured from a RGB-D camera. We use only the depth image to acquire the input cloud. The RGB image is used for visualization. For the output, besides the predicted pose, a clean hand is simultaneously reconstructed. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

the first frame, and sometimes tracking drift happens. Other conventional methods [OKA11b][BTG^{+}12][TBS^{+}16] resort to multi-camera setups to reduce the influence of object occlusions from multiple viewpoints. However, it is expensive and complex to set up a synchronous and calibrated system with multiple sensors.

Currently, hand pose estimation for hand-object interaction cases is limited by existing available datasets. Public large-scale datasets with reliable 3D ground-truth annotations are lacking due to the complexity of annotating 3D hand pose. Although some large-scale datasets, like *Hands2017Challenge* [YYGHK17], have accurate 3D pose annotations, they are entirely composed from clean hand samples. Therefore, it is worth considering how to utilize existing clean hand datasets for hand-object cases.

In this work, we propose a novel deep learning framework using Augmented Autoencoder to tackle hand-object interaction problem in hand pose estimation tasks. Our method takes 3D occluded hand point cloud as input, which is obtained by a random data augmentation process from clean hand samples. The encoder extracts point-wise features and fuses them to a latent vector. Addressing the problem of object occlusion in hand-object interaction cases, we use an auxiliary decoder to reconstruct the clean hand point cloud from the latent vector, and another decoder estimates simultaneously the 3D hand pose from it. To the best of our knowledge, this is the first work that uses 3D point cloud data to tackle object occlusion problem in hand-object interaction tasks (Fig. 1.2).

Our contribution can be summarized as follows:

- We present an augmentation strategy to simulate hand-object interaction cases utilizing existing large clean hand datasets. Since unlimited types of objects could be augmented, the trained model is more generalizable on unknown objects.

- We propose an auxiliary clean hand reconstruction decoder to improve the quality of the latent space, which in turn improves the hand pose accuracy.

- We demonstrate the advantages of the proposed augmentation and reconstruction approaches both qualitatively and quantitatively through multiple experiments.

Chapter 2

Related Work

In this chapter, we first review some hand pose estimation works on both clean hand and hand-object interaction cases. Then we briefly introduce the backbone of our framework, Augmented Autoencoder. Finally, we review some 3D point cloud reconstruction works, where the utilized 3D shape reconstruction method, FoldingNet, is introduced.

Clean Hand Pose Estimation

In the past few years, a lot of 2D deep learning based research for clean hand pose estimation has been done [WPVGY18][OWL15][WPVGY17][ZB17][SSPH18][YY19]. In particular, 2D depth image based methods demonstrate robust performance. Oberweger et al. [OWL15] use 2D CNN to estimate the hand pose from the image features, where they introduce a bottleneck layer to force the predicted pose obey certain prior distribution. Wan et al. [WPVGY18] estimate hand pose with a proposed pose parameterization strategy, which decomposes the pose parameters into a set of per-pixel estimations, i.e. 2D/3D heat maps and unit 3D directional vector fields, to leverage the 2D and 3D properties of the input depth map.

Recently, 3D deep learning methods gain more attention due to the abundant information in input data [LL19][QYSG17][GLYT16][GLYT17]. Ge et al. [GCWY18] present a PointNet [QYSG17] based approach that directly takes point clouds as input to regress 3D hand joint locations. In order to handle variations of hand global orientations, they introduce the oriented bounding box (OBB) to normalize the hand point clouds. Li et al. [LL19] propose a point-to-pose voting based residual permutation equivariant network for hand pose estimation task. Without the need of complex preprocessing steps, their method takes unordered 3D point cloud as input to compute point-wise features and through weighted fusion to obtain final hand pose estimates. Despite their good performance on hand pose estimation, they commonly ignore the crucial hand-object interaction cases.

Hand Pose Estimation with Object Interaction

There are some previous works that have taken the problem of object occlusion in hand pose estimation task into account [CHYCR17][TA18][ZBYX19][YK18][GWF+19] [MEC+17][TBP19].

The work by Tekin et al. [TBP19] has impressive success of 3D hand pose estimation jointly with other parallel tasks. Their method takes a sequence of frames as input and outputs per-frame 3D hand and object pose predictions along with the estimates of object and action categories for the entire sequence, whereas it relies too much on frame sequence rather than single image. Gao et al. [GWF+19] propose an object-aware method to estimate 3D hand pose from a single RGB image, where they rely on a deep structure to infer the category of the grasped object shape under the assumption that objects of a similar category are grasped in a similar way. Boukhayma et al. [BBT19] propose to use extracted hand parameters to control a mesh deformation hand model MANO [RTB17] and project it into image domain to train the network. A similar hand model based work by Hasson et al. [HVT+19] uses a contact loss to describe the spatial state of hand and object when a hand manipulates object, i.e. using a repulsion loss to penalize interpenetration and an attraction loss to encourage the hand to be in contact with the object. These methods require complex annotation process and could not fully utilize existing annotated clean hand datasets for hand-object interaction cases.

From Autoencoder to Augmented Autoencoder

Originally, the Autoencoder model introduced by Hinton et al. [RHW85] is used for dimensionality reduction for high dimensional data. The training objective of Autoencoder is to reconstruct the input after passing through a low dimensional space. With the success of deep learning networks, many variants [HYW+15][CKS+16][N+11] of Autoencoder emerged and have shown robust performance. Particularly, Vincent et al. [VLL+10] propose a Denoising Autoencoder to reconstruct denoised test data. Their strategy proposes to apply artificial random noise to input data while the reconstruction stays clean. Their work shows that the latent representation can be invariant to the insignificant input noise.

In 2018, Sundermeyer et al. proposed a real-time RGB-based pipeline for object detection and 6D pose estimation [SMD+18]. In their work, to remove the effects of object occlusions and background clutters, a random augmentation process with artificial occlusions and clutters is applied to the training data, by which they demonstrate that this training procedure enforces invariance not only against noise but also against a variety of different input augmentations. Encouraged by the idea of augmentation invariance, we apply a random augmentation process on clean hand samples of existing datasets to generate our input, and recover corresponding clean hand samples with an auxiliary 3D shape reconstruction decoder.

3D Shape Reconstruction

3D Shape Reconstruction using deep learning has made a lot of advancement in recent years [SUHR17][CXG$^+$16][GFK$^+$18][YFST18][FSG17]. Fan et al. [FSG17] propose a conditional shape sampler, capable of predicting multiple plausible 3D point clouds from the input pair of a 2D image and an additional random vector, which is used to perturb the prediction from the image. Yang et al. [YFST18] propose a folding-based network, FoldingNet, which deforms a canonical 2D grid onto the underlying 3D target surface of a point cloud with two consecutive folding operations. FoldingNet consumes only about 7% parameters of a fully-connected layer based neural network to reconstruct a 3D target. Their method achieves low reconstruction errors even for targets with delicate structures. Groueix et al. propose a shape generation framework, AtlasNet [GFK$^+$18]. Compared with FoldingNet, AtlasNet represents a 3D shape as a collection of multiple parametric surface elements instead of a single surface element. Their method achieves state-of-the-art reconstruction performance with multiple patches.

A critical challenge in 3D shape reconstruction is to evaluate the predicted point cloud. The loss function should be not only computationally efficient but also differentiable with respect to point coordinates. The Chamfer Distance (CD) and the Earth Mover's Distance (EMD) [RTG00] are two outstanding candidates to compare the reconstructed clean hand point cloud with ground-truth in our work.

Chapter 3

Methodology

In this chapter, the technical approaches of our method are presented. Firstly, we give an overview of the proposed framework. Then, the detailed data processing approach is introduced. Finally, in the implementation subsection, we introduce the theoretical models behind our model, based on which we show how we implement our networks, such as the encoder, the clean hand reconstruction decoder and the pose estimation decoder. Additionally, the corresponding training losses are expanded in details.

3.1 Overview

The framework of our method is illustrated in Fig. 3.1. For visual convenience of 3D points, all points are painted with color of different levels of brightness with respect to the distance to camera. That is, a point is darker when it is more distanced, and vice versa. In the later sections, we obey this rule by default.

As depicted, our method takes an occluded hand point cloud as input, which is generated by a random augmentation process. For the encoder, we use a residual network version of Permutation Equivariant Layer (PEL) to extract point-wise features and a voting-based scheme [LL18] followed by a fully-connected layer to merge valuable information from individual point to final Gaussian distributed latent vector. Then, the acquired latent vector is used to reconstruct clean hand point cloud and estimate hand pose by the decoder side. The auxiliary Decoder 1 is a folding-based deep network named FoldingNet [YFST18], which expressively folds out the reconstructed clean hand point cloud, while Decoder 2 estimates hand pose with a deep fully-connected network. Alternatively, we also try an AtlasNet [GFK$^+$18] based reconstruction decoder for the auxiliary Decoder 1. Correspondingly, they have their own loss functions, reconstruction loss and pose loss. Since this architecture is based on the VAE [KW14] theory, a KL loss for latent variables is also crucial in our method.

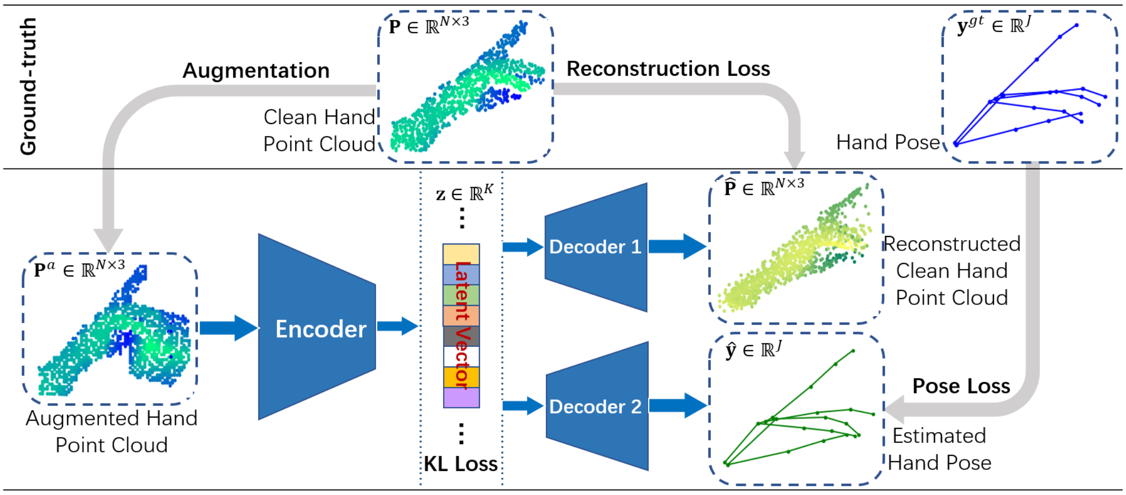

Ground-truth
Augmentation
$\mathbf{P} \in \mathbb{R}^{N\times3}$
Clean Hand Point Cloud
Reconstruction Loss
$\mathbf{y}^{gt} \in \mathbb{R}^{J}$
Hand Pose
$\mathbf{P}^{a} \in \mathbb{R}^{N\times3}$
Augmented Hand Point Cloud
Encoder
$\mathbf{z} \in \mathbb{R}^{K}$
Latent Vector
KL Loss
Decoder 1
Decoder 2
$\hat{\mathbf{P}} \in \mathbb{R}^{N\times3}$
Reconstructed Clean Hand Point Cloud
$\hat{\mathbf{y}} \in \mathbb{R}^{J}$
Pose Loss
Estimated Hand Pose

Figure 3.1: Overview of our method. The input of our network is an occluded hand point cloud, which is obtained by a random augmentation process from a clean hand point cloud. The encoder encodes the input hand to a latent vector. The obtained latent vector is then used to reconstruct clean hand point cloud by the auxiliary Decoder 1 and predict 3D hand pose by Decoder 2. There are three losses in our VAE based framework, which are the KL loss, reconstruction loss and pose loss. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

3.2 Data Processing

For data processing, two significant approaches are the view normalization (Section 3.2.1) of the hand point clouds obtained from depth images and the data augmentation process (Section 3.2.2) for clean hand point clouds.

3.2.1 View Normalization

For pre-processing of the data frames, firstly, the depth pixels in the raw depth images are converted to 3D points. Then, a 3D bounding box is created for the hand points to obtain normalized coordinates of these points. The commonly used method will simply create a bounding box aligned with the camera coordinate system. However, this will lead to different set of observation points for the exact same pose label, which causes one-to-many mapping of the input-output pairs (Fig. 3.2a).

In order to maintain the one-to-one mapping relation of the input-output pairs, we use a view normalization process on each hand point cloud to align the the bounding box's z-axis $[0, 0, 1]^T$ with the view direction towards the hand centroid point $\mathbf{c} \in \mathbb{R}^3$. The alignment is performed by rotating the hand points with the rotation matrix $\mathbf{R}_{cam}$:

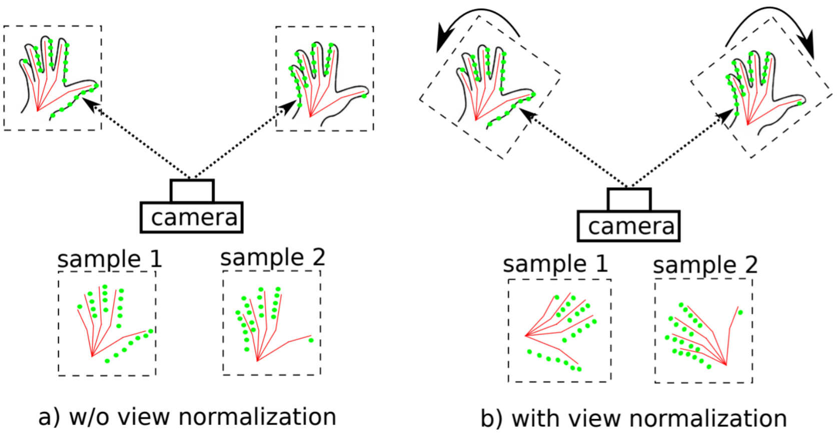


Figure 3.2: View normalization for hand point clouds. Red skeletons indicate ground-truth pose, green points indicate observed points of the camera. a) Due to different view directions, different observations lead to the same hand pose, thus the resulted training samples will contain one-to-many mappings. b) With view normalization, the different observations will also have different pose labels, thus the input-output pairs have the one-to-one mapping relation. [LL18]

$$\begin{aligned}
\alpha_y &= atan2(\mathbf{c}_x, \mathbf{c}_z), \\
\widetilde{\mathbf{c}} &= \mathbf{R}_y(-\alpha_y) \cdot \mathbf{c}, \\
\alpha_x &= atan2(\widetilde{\mathbf{c}}_y, \widetilde{\mathbf{c}}_z), \\
\mathbf{R}_{cam} &= \mathbf{R}_y(-\alpha_y) \cdot \mathbf{R}_x(\alpha_x),
\end{aligned} \tag{3.1}$$

where $\mathbf{R}_x$, $\mathbf{R}_y$ are the rotation matrices around the x-axis and y-axis respectively. If the rotation angle is θ, $\mathbf{R}_x$, $\mathbf{R}_y$ are defined as:

$$\begin{aligned}
\mathbf{R}_x(\theta) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}, \\
\mathbf{R}_y(\theta) &= \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}.
\end{aligned} \tag{3.2}$$

After rotating the observation points and ground truth pose with the rotation matrix $\mathbf{R}_{cam}$, the hand is rotated such that it appears right in front of the camera. As illustrated in Fig. 3.2b, the one-to-many mapping problem is avoided.

3.2.2 Data Augmentation

The motivation behind our Augmented Autoencoder based hand pose estimation framework is to control what the latent vector encodes and which properties are ignored. To take advantages of current large-scale clean hand dataset, we apply

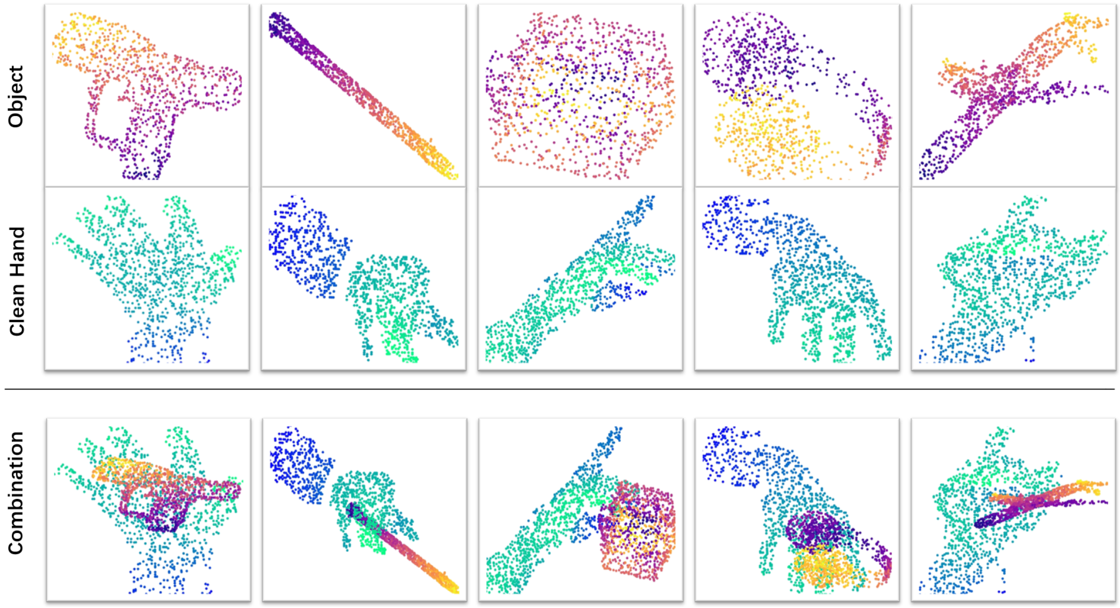


Figure 3.3: Combination results. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

a random augmentation process by superimposing random objects from *ShapeNet* [CFG$^+$15] on clean hands to simulate hand-object interaction scenarios in reality. Simultaneously, the clean hand point cloud also serves as the ground-truth for reconstructed points by the auxiliary Decoder 1. Through this approach, we make the latent representation invariant against object occlusions when a hand is in contact with an object.

Specifically, the augmentation work contains mainly three steps, which are ”combine hand point cloud and object”, ”project combined point cloud to depth image” and ”project depth image to occluded hand point cloud”.

Combine hand point cloud and object

For the variability of training samples, we randomly select a normalized object point cloud from the preprocessed *ShapeNet* dataset. In addition to this, we perform some basic transformations, i.e. random rotation, scaling and translation on the object. The random scaling operation resizes the object to the comparative size with a hand. Then, with a random translation, we shift the object around the average center of all joints to simulate real case when people grab an object.

For the rotation operation, firstly, we sample three random angles θ_x, θ_y, θ_z within $[-\pi, \pi]$ *radian*. Then the object points are rotated by θ_x, θ_y and θ_z around the x-, y- and z-axis respectively using the rotation matrix $\mathbf{R}$:

$$\mathbf{R} = \mathbf{R}_x(\theta_x) \cdot \mathbf{R}_y(\theta_y) \cdot \mathbf{R}_z(\theta_z), \tag{3.3}$$

where $\mathbf{R}_x$, $\mathbf{R}_y$ are defined in Equation 3.2, and the rotation matrix $\mathbf{R}_z$ is defined as:

$$\mathbf{R}_z(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.4}$$

For each point $\mathbf{p} = [x, y, z]^T$ in the object point cloud, the obtained point $\mathbf{p}' = [x', y', z']^T$ after the basic transformations is:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \gamma \mathbf{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{t}, \tag{3.5}$$

where γ is the scaling factor and $\mathbf{t} = [t_x, t_y, t_z]^T$ denotes the random translation. The last step is to superimpose the object on a hand sample. Some combination results are depicted in Fig. 3.3.

Project combined point cloud to depth image

Generally, 3D data are collected in the form of depth image by various cameras in reality. Therefore, we next render the combination results to depth images, where we only keep the point which is the closest to the camera among those projected to the same 2D image grid. Meanwhile, in order to acquire good occlusion quality, we take three measures. First, we sample as many points as possible from the 3D meshes in *ShapeNet* during the previous pre-processing. Second, within an appropriate range, we make the distance of the camera from the point cloud far enough during projection. The last one is that we adjust the size of pixel grid to a suitable value, which is 80×80 depth image.

For each 3D point $[x, y, z]^T$, the location of the corresponding pixel on the depth image is in column u and row v with the value d:

$$\begin{aligned} u &= \frac{x}{z} \cdot f_x + u_0, \\ v &= \frac{y}{z} \cdot f_y + v_0, \\ d &= z, \end{aligned} \tag{3.6}$$

where u_0, v_0, f_x, f_y are the intrinsic parameters of the depth camera. The transformed depth images corresponding to the combinations in Fig. 3.3 are illustrated in Fig. 3.4. As mentioned before (Section 3.1), brightness indicates depth value. Brighter pixel represents smaller depth value, and vice versa.

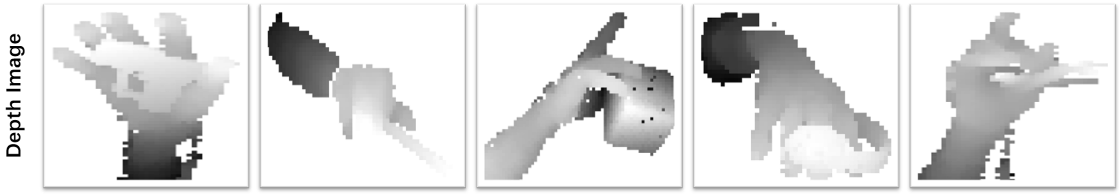


Figure 3.4: Project combination results to depth images. (Brightness indicates depth, i.e. darker denotes further.)

Project depth image to occluded hand point cloud

For the last step of data augmentation, we convert depth images to occluded hand point clouds, which is the inverse transformation of Equation 3.6:

$$\begin{aligned} x &= \frac{z}{f_x} \cdot (u - u_0), \\ y &= \frac{z}{f_y} \cdot (v - v_0), \\ z &= d, \end{aligned} \tag{3.7}$$

Fig. 3.5 shows the occluded hand point clouds corresponding to those depth images in Fig. 3.4.

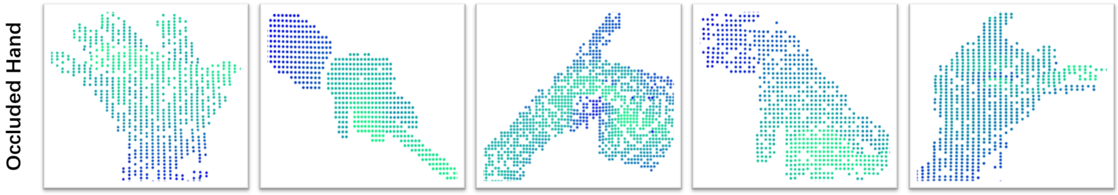


Figure 3.5: Project depth images to final occluded hand point clouds. Brightness indicates depth value, namely, darker point denotes that the point is more distanced to camera, and vice versa.

3.3 Implementation

As depicted in Fig. 3.1, the whole framework consists of a clean hand point cloud reconstruction network and a hand pose estimation pipeline. In this section, we will firstly introduce the theoretical models behind our method, Augmented Autoencoder (Section 3.3.1) and Variational Autoencoder (Section 3.3.2). Then, all invoked components in this network are elaborate in respective subsections, such as Residual Permutation Equivariant Layer based Encoder (Section 3.3.3), Folding-based Decoder for Points Reconstruction (Section 3.3.4), and Fully-connected Layer based Decoder for Pose Estimation (Section 3.3.5). Furthermore, three training losses in our method are expanded in details in Section 3.3.6.

3.3.1 Augmented Autoencoder

Augmented Autoencoder (AAE) serves as the backbone of our method, which is from the work *Implicit 3D Orientation Learning for 6D Object Detection from RGB Images* [SMD+18] by Sundermeyer et al..

In the first part of this work, an augmented training model is proposed to remove the effects of object occlusions and background clutter. The training architecture is illustrated in Fig. 3.6. The target batch is reconstructed after the augmented input passing through a low-dimensional bottleneck, referred to as the latent representation. And the loss function is simply a sum over the pixel-wise L2 distance between the reconstruction result and the original training data.

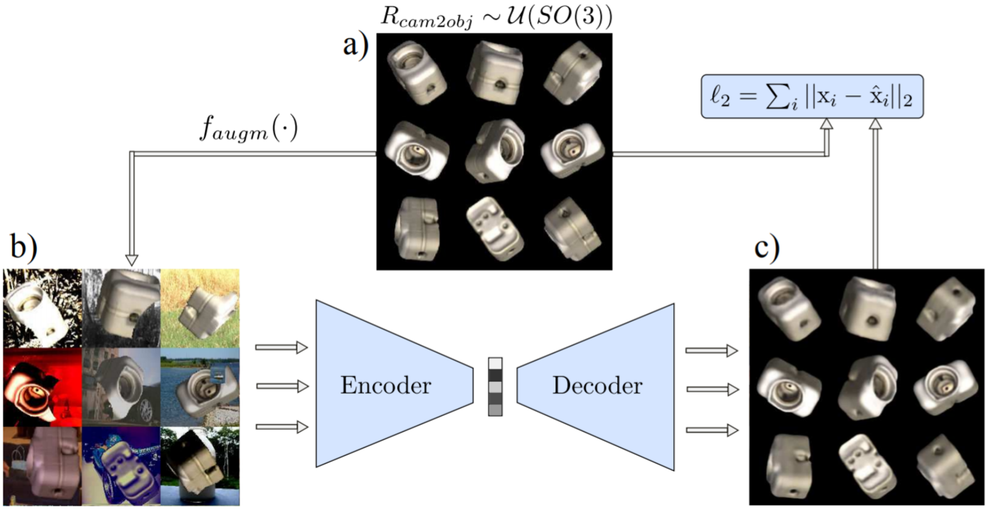

$R_{cam2obj} \sim \mathcal{U}(SO(3))$
a)
$\ell_2 = \sum_i ||\mathrm{x}_i - \hat{\mathrm{x}}_i||_2$
$f_{augm}(\cdot)$
b)
c)
Encoder
Decoder

Figure 3.6: Training process for the AAE. a) reconstruction target batch of uniformly sampled SO(3) object views; b) geometric and color augmented input; c) reconstruction results. [SMD+18]

Actually, the motivation behind the AAE is to control what the latent representation encodes and which properties are ignored. In this work, random augmentation is applied to the input images against which the encoding shall be invariant. Sundermeyer et al. successfully confirm that this training strategy produces latent representations which are able to be invariant to a variety of different input augmentations. Encouraged by this, we introduce this strategy to our framework by the data augmentation process in Section 3.2, to make the encodings invariant to the insignificant objects.

3.3.2 Variational Autoencoder

In our work, we have a modified training procedure for the AAE. Considering the excellent adaptability and generative ability of Variational Autoencoder (VAE) [KW14] to various data, we construct our network based on the theoretical model of VAE.

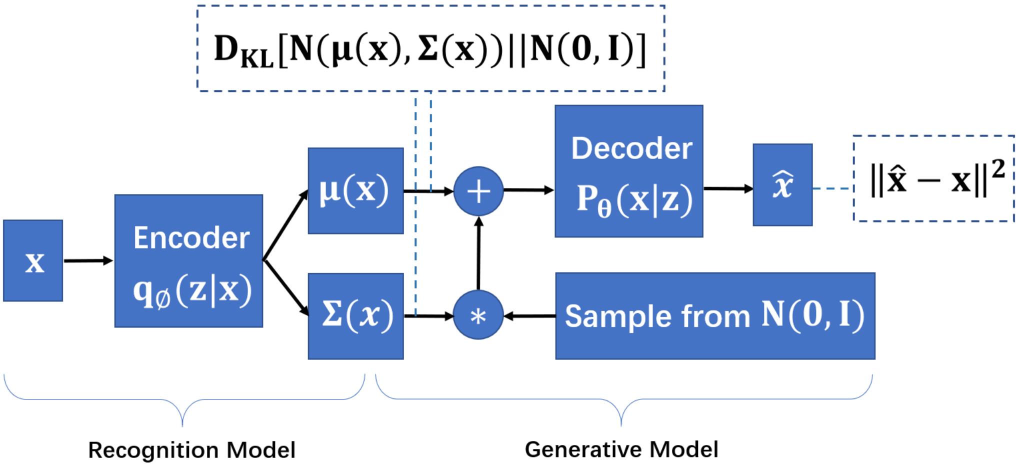


Figure 3.7: A training-time VAE with reparameterization trick. Dash line box shows loss layers, $\boldsymbol{\mu}(\mathbf{x})$ and $\boldsymbol{\Sigma}(\mathbf{x})$ denote the mean and the variance of the latent variable $\mathbf{z}$, ϕ and θ signify model parameters.

Fig. 3.7 shows the probabilistic model of a training-time VAE. The VAE framework consists of a recognition model $\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$ and a generative model $\mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$, where $\mathbf{x}$ is the observed random variable, $\mathbf{z}$ denotes latent variable, ϕ and θ signify the respective model parameters. Here, since the true posterior $\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})$ is intractable, it invokes the variational approximate posterior $\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$ to be multivariate Gaussian with a diagonal covariance structure. The probabilistic encoder $\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$ produces a distribution over the possible values of the code $\mathbf{z}$ given data $\mathbf{x}$. Note that the latent variable $\mathbf{z}$ is assumed to be the centered isotropic multivariate Gaussian, i.e. $\mathbf{p}_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, \mathbf{I})$. And given code $\mathbf{z}$ the probabilistic decoder $\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})$ produces a distribution over the corresponding values of $\mathbf{x}$.

The optimization objective for VAE is to maximize the variational lower bound $\mathcal{L}(\phi, \theta; \mathbf{x})$, which is the sum of a KL divergence term $-\mathbf{D_{KL}}(\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})||\mathbf{p}_\theta(\mathbf{z}))$ and a negative reconstruction error term in autoencoder parlance. The KL divergence term can be interpreted as regularizing ϕ, encouraging the approximate posterior to be close to the prior $\mathbf{p}_\theta(\mathbf{z})$.

VAE model exhibits excellent performance in implementing efficient approximate posterior inference of the latent variable given an observed value and allows us to perform all kinds of inference tasks where a prior over observed data is required. Therefore, it is an outstanding solution for our use case, i.e. to remove the effects of object occlusions.

3.3.3 Residual Permutation Equivariant Layer based Encoder

The encoder in our method is a modified structure based on Residual Permutation Equivariant Layer (PEL) in *Point-to-pose Voting based Hand Pose Estimation using Residual Permutation Equivariant Layer* [LL18]. In this paper, Li et al. utilize PEL as basic element for a deep network to extract point-wise features of an unordered point cloud and merge these features with a novel point-to-pose voting scheme to final hand pose. Considering the similarity of the hand pose estimation problem, we invoke this method as part of our encoder components to extract latent variables of the hidden random process.

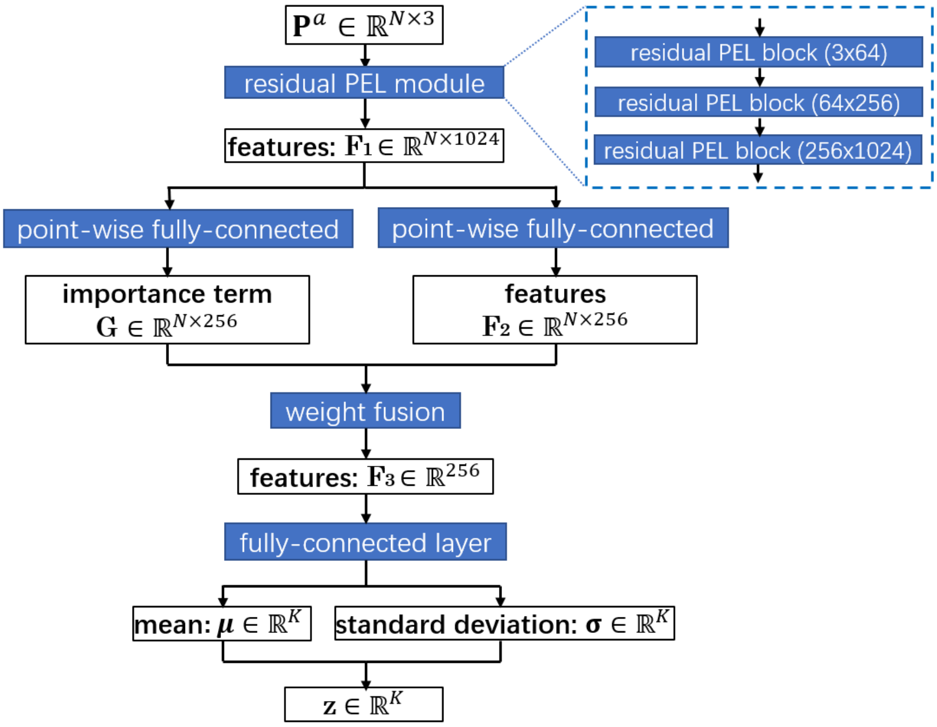

$\mathbf{P}^a \in \mathbb{R}^{N\times3}$
residual PEL module
residual PEL block (3x64)
residual PEL block (64x256)
residual PEL block (256x1024)
features: $\mathbf{F}_1 \in \mathbb{R}^{N\times1024}$
point-wise fully-connected
point-wise fully-connected
importance term $\mathbf{G} \in \mathbb{R}^{N\times256}$
features $\mathbf{F}_2 \in \mathbb{R}^{N\times256}$
weight fusion
features: $\mathbf{F}_3 \in \mathbb{R}^{256}$
fully-connected layer
mean: $\boldsymbol{\mu} \in \mathbb{R}^K$
standard deviation: $\boldsymbol{\sigma} \in \mathbb{R}^K$
$\mathbf{z} \in \mathbb{R}^K$

Figure 3.8: Residual PEL based encoder network. K denotes the number of dimensions of latent variables, dash line box describes the detailed structure for one of the residual blocks.

As illustrated in Fig. 3.8, the encoder in our method is a modified structure based on Residual Permutation Equivariant Layer (PEL) [LL19]. The input occluded hand point cloud $\mathbf{P}^a \in \mathbb{R}^{N\times3}$ represented by N unoredered 3D points passes firstly through a residual PEL module, which consists of 3 residual PEL blocks. Then point-wise feature $\mathbf{F_1} \in \mathbb{R}^{N\times1024}$ is computed for each individual input point, where each row of $\mathbf{F_1}$ represents the local feature for one point. The obtained $\mathbf{F_1}$ is imported to two separate point-wise fully-connected modules respectively. Correspondingly, two separate terms are computed, an importance term $\mathbf{G} \in \mathbb{R}^{N\times256}$ and a new feature term $\mathbf{F_2} \in \mathbb{R}^{N\times256}$, where the local feature dimension for each point is shrunk to 256. Each element of $\mathbf{G}$ indicates the weight for corresponding feature value in $\mathbf{F_2}$ and provides vital information of the importance of current feature value. Then, by a weight fusion module, we merge the information of both terms to $\mathbf{F_3} \in \mathbb{R}^{256}$:

$$\mathbf{f}_i = \frac{\sum_{n=1}^{N}(\mathbf{G}_{ni}\mathbf{F_2}_{ni})}{\sum_{n=1}^{N}\mathbf{G}_{ni}}, \tag{3.8}$$

where $\mathbf{f}_i$ is the i-th feature value in $\mathbf{F_3}$.

In order to extract complex features, we use a 5-layer perceptron to encode $\mathbf{F_3}$ to the final K-dimensional latent vector, which consists of a latent mean vector $\boldsymbol{\mu} \in \mathbb{R}^K$, and a latent standard deviation vector $\boldsymbol{\sigma} \in \mathbb{R}^K$.

During training stage, a reparameterization process to sample from the distribution of the latent vector [KW13] is needed:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \tag{3.9}$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^K$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ denotes element-wise multiplication. The final latent vector $\mathbf{z} \in \mathbb{R}^K$ is Gaussian distributed and $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$.

3.3.4 Decoders for Points Reconstruction

For clean hand point cloud reconstruction, we have two candidates, which are FoldingNet and AtlasNet, respectively. In this section, the modified versions of both approaches are introduced.

FoldingNet based Decoder

The folding based decoding operation proposed by Yang et al. in *FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation* [YFST18] is strongly expressive and universal in constructing point clouds. In our work, we use a modified version of the folding-based method to implement 3D clean hand point cloud reconstruction for Decoder 1. The architecture of this deep decoder network is illustrated in Fig. 3.9.

The obtained latent vector $\mathbf{z}$ from encoder is fed into Decoder 1. Assume that we use N points to represent the reconstructed clean hand point cloud $\hat{\mathbf{P}} \in \mathbb{R}^{N\times3}$, besides

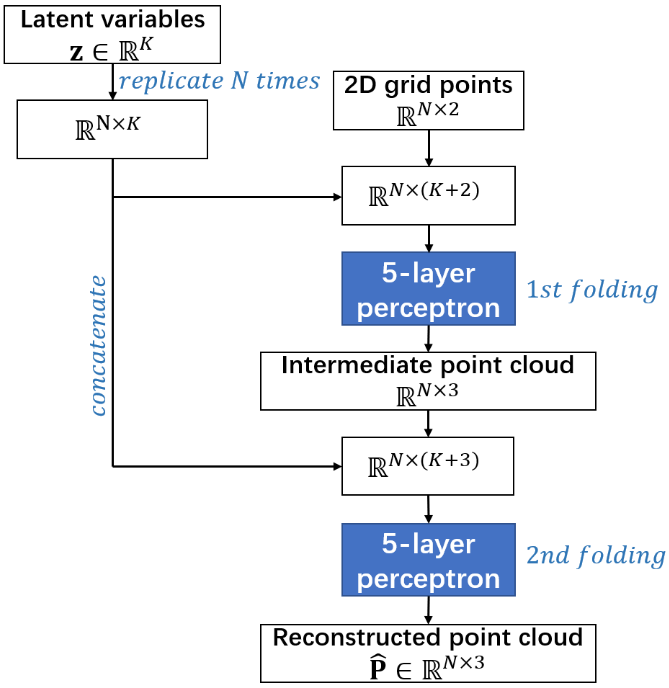


Figure 3.9: FoldingNet based decoder network for points reconstruction. K denotes the number of dimensions of latent variables.

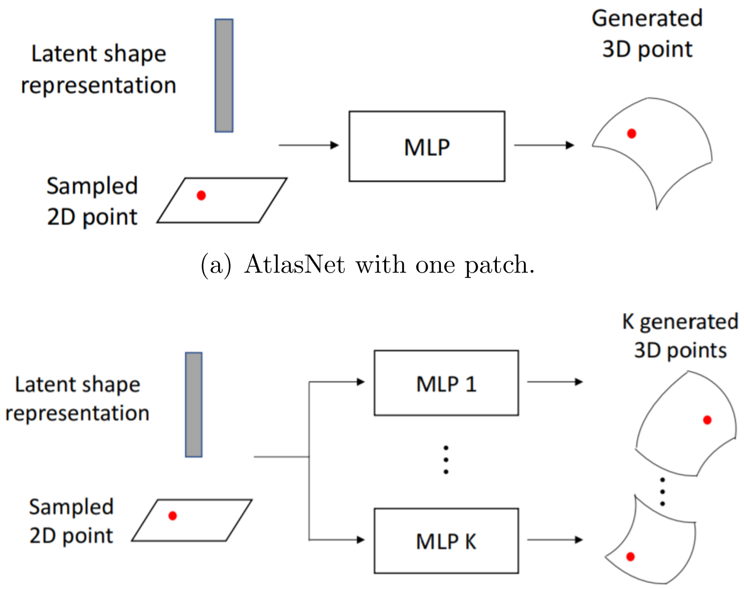
$\mathbf{z}$, N fixed 2D grid points uniformly sampled from a unit square are needed. Additionally, the latent vector $\mathbf{z}$ is replicated N times. Totally, we perform 2 consecutive folding operations in this decoder network. The 1st folding is started by concatenating the 2D grid points to the replicated latent variables, and the 2nd folding concatenates the intermediate point cloud from 1st folding to the replicated latent variables again. We call the concatenated results feature vectors. Both the feature vectors from concatenation are then processed through a 5-layer perceptron. Each perceptron layer is independently applied to the feature vector of a single point, i.e. each row of the feature vectors.

After two consecutive folding operation, the reconstructed clean hand point cloud $\hat{\mathbf{P}}$ is produced.

AtlasNet based Decoder

AtlasNet proposed by Groueix et al. [GFK+18] represents a 3D shape as a collection of parametric surface elements, which is obtained from a set of 2D squares. The model is illustrated in Fig. 3.10.

For the approach with one batch, their method is actually similar to the architecture of FoldingNet with one time folding operation. Besides the latent shape representation, the approach takes a set of 2D grid points as additional input, which are uniformly sampled in the unit square. These points are used to generate points on the 3D shape surface (Fig. 3.10(a)). This approach can be repeated multiple times to represent a 3D shape as the union of several surface elements (Fig. 3.10(b)). AtlasNet achieves state-of-the-art performance with multiple patches.



(a) AtlasNet with one patch.

(b) AtlasNet with K patches.

Figure 3.10: The shape generation approach for AtlasNet. [GFK+18]

In our work, we use a modified version of AtlaNet with 5 patches to reconstruct the clean hand point clouds for Decoder 1. The architecture of this deep network is illustrated in Fig. 3.11.

The obtained latent vector $\mathbf{z}$ from encoder is fed into Decoder 1. Assume that we use N points to represent the reconstructed clean hand point cloud $\hat{\mathbf{P}} \in \mathbb{R}^{N\times 3}$, besides $\mathbf{z}$, $N/5$ fixed 2D grid points uniformly sampled from a unit square are needed. For each patch, the latent vector $\mathbf{z}$ is replicated $N/5$ times and then concatenated with the grid points to obtain the feature vectors. Then, the feature vectors from concatenation are processed through a 5-layer perceptron. Each perceptron layer is independently applied to the feature vector of a single point, i.e. each row of the feature vectors. After 5 parallel pipelines for these feature vectors, 5 patches are acquired and combined to the final reconstructed clean hand point cloud $\hat{\mathbf{P}}$.

Note that, in order to compare the two candidates for clean hand points reconstruction, for the AtlasNet based decoder we use comparative number of parameters to the FoldingNet based decoder. As we can see from the structure of the Atlas-

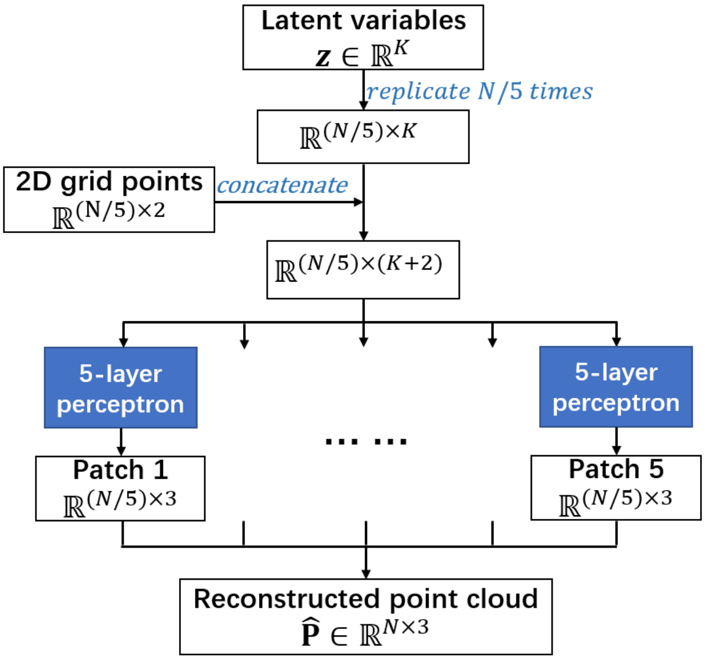

Latent variables
$z \in \mathbb{R}^K$
replicate $N/5$ times
$\mathbb{R}^{(N/5)\times K}$
2D grid points
$\mathbb{R}^{(N/5)\times 2}$
concatenate
$\mathbb{R}^{(N/5)\times(K+2)}$
5-layer perceptron
5-layer perceptron
… …
Patch 1
$\mathbb{R}^{(N/5)\times 3}$
Patch 5
$\mathbb{R}^{(N/5)\times 3}$
Reconstructed point cloud
$\hat{\mathbf{P}} \in \mathbb{R}^{N\times 3}$

Figure 3.11: AtlasNet based decoder network for points reconstruction. K denotes the number of dimensions of latent variables.

Net based decoder network, 3 more 5-layer perceptrons are used compared to the FoldingNet based decoder.

3.3.5 Fully-connected Layer based Decoder for Pose Estimation

For 3D hand pose prediction, Decoder 2, which consists of 5 fully-connected layers, takes the reparameterized latent vector $\mathbf{z}$ as input and outputs the vectorized 3D hand pose $\hat{\mathbf{y}} \in \mathbb{R}^J$, where $J = 3 \times \#joints$. Since we present human hand using 21 joints, the output 3D hand pose is in a vectorized form $\mathbb{R}^{63}$, and $\#joints = 21$.

3.3.6 Training Loss

As mentioned in Section 3.1, there are three training loss in our method, which are Reconstruction Loss, Pose Loss and KL Loss, respectively.

Reconstruction Loss

The design of loss function for comparing the reconstructed clean hand point cloud $\hat{\mathbf{P}} \in \mathbb{R}^{N\times 3}$ and the ground truth $\mathbf{P} \in \mathbb{R}^{N\times 3}$ is crucial for points reconstruction. The basic requirement for the reconstruction loss function is differentiability with respect to point locations. Since data will be forward and backward propagated for many times, it must be efficient to compute. Furthermore, it should be robust enough against small number of outlier points in the sets.

In our work, we have two loss functions to evaluate the points reconstruction results: Chamfer Distance (CD) and Earth Mover's Distance (EMD) [FSG17].

The Chamfer Distance is defined as:

$$\mathcal{L}_{CD}\left(\hat{\mathbf{P}},\mathbf{P}\right)=\frac{1}{|\hat{\mathbf{P}}|}\sum_{\hat{p}\in\hat{\mathbf{P}}}\min_{p\in\mathbf{P}}\|\hat{p}-p\|+\frac{1}{|\mathbf{P}|}\sum_{p\in\mathbf{P}}\min_{\hat{p}\in\hat{\mathbf{P}}}\|\hat{p}-p\|, \tag{3.10}$$

where the CD algorithm finds for each point the nearest neighbor in the other point cloud and sums up the Euclidean distances.

The Earth Mover's Distance requires that $\hat{\mathbf{P}}$ and $\mathbf{P}$ have the same size, i.e. $|\hat{\mathbf{P}}|=|\mathbf{P}|$, and it is defined as:

$$\mathcal{L}_{EMD}\left(\hat{\mathbf{P}},\mathbf{P}\right)=\frac{1}{|\mathbf{P}|}\min_{\phi:\mathbf{P}\rightarrow\hat{\mathbf{P}}}\sum_{p\in\mathbf{P}}\|p-\phi\left(p\right)\|, \tag{3.11}$$

where ϕ denotes one-to-one bijective correspondences from the ground-truth $\mathbf{P}$ to the predicted point set $\hat{\mathbf{P}}$. The Euclidean distances of all matched point pairs are then summed.

Both loss functions have their own intrinsic characteristics. For example, while EMD roughly captures the shape corresponding to the mean value of the hidden variable of the hand point cloud, CD tends to give a splashy shape that blurs the shape's geometric structure [FSG17]. To make the reconstruction by Decoder 1 more expressive, we combine both loss functions during training time. Therefore, implicitly, our method requires the reconstructed clean hand points have the same size N as the ground-truth.

Pose Loss

The training loss for hand pose is simply the L2 loss between the predicted pose $\hat{\mathbf{y}}$ and the ground truth pose $\mathbf{y}^{gt}\in\mathbb{R}^{J}$.

The L2 loss is defined as:

$$\mathcal{L}_{pose}=\frac{1}{2}\sum_{j=1}^{J}\left(\hat{\mathbf{y}}_j-\mathbf{y}_j^{gt}\right)^2, \tag{3.12}$$

where J is the number of dimensions of the hand pose vector, i.e. $J=3\times\#joints$. Since we present human hand using 21 joints, the output 3D hand pose is in a vectorized form $\mathbb{R}^{63}$, and $\#joints=21$.

KL Loss

KL loss is also called KL divergence in the variational lower bound. Based on the VAE theory, a KL loss is essential to force the computed latent vector $\mathbf{z}$ given observed occluded data to be close to the centered isotropic multivariate Gaussian $\mathcal{N}\left(\mathbf{z};\mathbf{0},\mathbf{I}\right)$.

The KL loss is defined as:

$$\mathcal{L}_{KL} = \frac{1}{2}\sum_{k=1}^{K}\left(\mu_k^2 + \sigma_k^2 - \log\left(\sigma_k^2\right) - 1\right), \tag{3.13}$$

where K denotes the number of dimensions of the latent vector $\mathbf{z}$, μ_k is the k-th dimension of the latent mean $\boldsymbol{\mu}$ and σ_k denotes the k-th dimension of the latent standard deviation $\boldsymbol{\sigma}$.

The resulting total loss for our method is the summation of $\mathcal{L}_{CD}$, $\mathcal{L}_{EMD}$, $\mathcal{L}_{pose}$ and weighted $\mathcal{L}_{KL}$ terms:

$$\mathcal{L}_{total} = \mathcal{L}_{CD} + \mathcal{L}_{EMD} + \mathcal{L}_{pose} + \alpha\mathcal{L}_{KL}, \tag{3.14}$$

where α is the weighting factor.

Chapter 4

Experiment and Evaluation

In this chapter, we firstly introduce the training details of our method where the parameter configuration is given. Then, the used datasets are described. Based on these datasets we present different experiments and evaluate the performance of different methods both qualitatively and quantitatively, where different evaluation metrics are used.

4.1 Training Details

Our method is implemented using the TensorFlow framework [ABC$^+$16] with the ADAM optimizer. The learning rate is tapered down from 0.01 to 0.00001 during the course of training. For all experiments, we use an input and reconstruction point size of $N = 625$ for training, and $N = 900$ for testing. For the latent vector $\mathbf{z} \in \mathbb{R}^K$, we set the number of dimension $K = 64$ and the KL Loss is weighted using a factor of $\alpha = 0.001$. Before our object augmentation process, we perform for each hand sample random translation in all three dimensions within $[-15, 15]$ mm, random scaling within $[0.75, 1.25]$ and random rotation around z-axis within $[-\pi, \pi]$ $radian$.

4.2 Datasets

For training and evaluating the proposed network, we use the *Hands2017Challenge* dataset [YYGHK17], the *SynthHands* dataset [MMS$^+$17] and also the *EgoDexter* dataset [MMS$^+$17].

As introduced previously, the input data of our framework are artificial occluded hand point clouds, which are acquired by combination of a random clean hand point cloud from the existing datasets (*Hands2017Challenge*, *SynthHands*) and an arbitrary object from the repository *ShapeNet* [CFG$^+$15]. We use the *EgoDexter* dataset to compare our method with the state-of-the-art method in [MMS$^+$17]. For data augmentation, we use the object repository *ShapeNet*. This section presents some meaningful details about these data collections and explains some pre-processing on them.

Hands2017Challenge

The *Hands2017Challenge* dataset is collected from parts of the *BigHand2.2M* [YYS$^+$17] and the *First-Person Hand Action (FHAD)* [GHYBK18]. The training set contains 957032 depth images, and the test set contains 295510 depth images. All samples in *Hands2017Challenge* are clean hands, where the hands are not in contact with objects.

The dataset is fully annotated (21-joints) using an automatic annotating system with 6D magnetic sensors and inverse kinematics. The depth images are captured with the latest Intel RealSense SR300 camera at 640×480-pixel resolution. This dataset has accurate annotations and exhibits a significantly wider and denser range of hand poses.

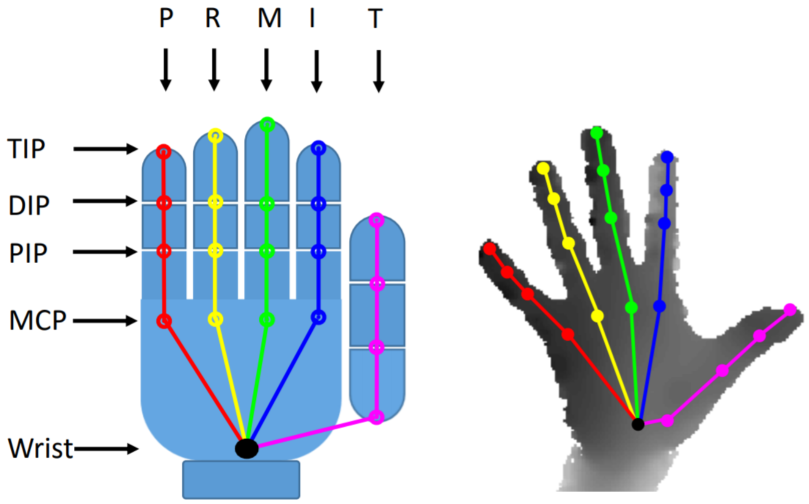


Figure 4.1: Joint annotation [YYGHK17]

The joint annotations for each hand image follow the following format: [Wrist, TMCP, IMCP, MMCP, RMCP, PMCP, TPIP, TDIP, TTIP, IPIP, IDIP, ITIP, MPIP, MDIP, MTIP, RPIP, RDIP, RTIP, PPIP, PDIP, PTIP], where 'T', 'I', 'M', 'R', 'P' denote 'Thumb', 'Index', 'Middle', 'Ring', 'Pinky' fingers. 'MCP', 'PIP', 'DIP', 'TIP' are joints' names as shown in Fig. 4.1. In all experiments of this work, we will follow this format of joint annotation.

Since our network takes unordered points as input, first and foremost, we perform a projection to transform the raw depth images to clean hand point clouds.

SynthHands

The egocentric dataset *SynthHands* is a synthetic dataset created by posing a photorealistic hand model with real hand motion data. It captures multiple variations in natural hand motion, such as pose, skin color, shape, texture, background clutter as well as camera viewpoint.

This dataset contains accurate annotated 92536 RGB-D images of clean hands and 91600 RGB-D images of hands interacting with objects, of which we use 69402 clean samples and 68700 interacting hand samples for training. Except the training samples, the rest 23134 clean samples serve as our *clean test set* and 22900 interacting samples as our *interacting test set*.

EgoDexter

The benchmark dataset *EgoDexter* consists of 3190 frames of natural hand interactions with objects in real cluttered scenes, moving egocentric viewpoints, complex hand-object interactions and natural lighting. In total 4 sequences are gathered (Rotunda, Desk, Kitchen, Fruits) featuring 4 different users (2 female), skin color variation, background variation, different objects and camera motion. Of these, 1485 frames are annotated with 3D finger tip positions.

We compare the 3D pose accuracy of our method to the state-of-the-art method in [MMS⁺17] using this dataset. Furthermore, we exclude the Kitchen sequence due to its many annotation errors, and use the other three sequences for evaluation.

ShapeNet

For the random augmentation process for clean hand samples, we use objects from *ShapeNetCore*, which is a subset of the object repository *ShapeNet* [CFG⁺15] and covers 55 object categories with about 51300 unique 3D models.

All 3D CAD shapes in this repository are presented in the format of '.obj'. With an offline preprocessing, we sample these 3D meshes to point clouds in the format of '.pcd' using pcl-tools, as illustrated in Fig. 4.2. After this, we perform centralization to set the geometrical center of each object at the origin. And the last operation of this preprocessing is normalization for each object point cloud.

4.3 Evaluation Metrics

To identify the success and failure modes of different approaches, different error metrics are used to evaluate the hand pose results. In our work, we evaluate

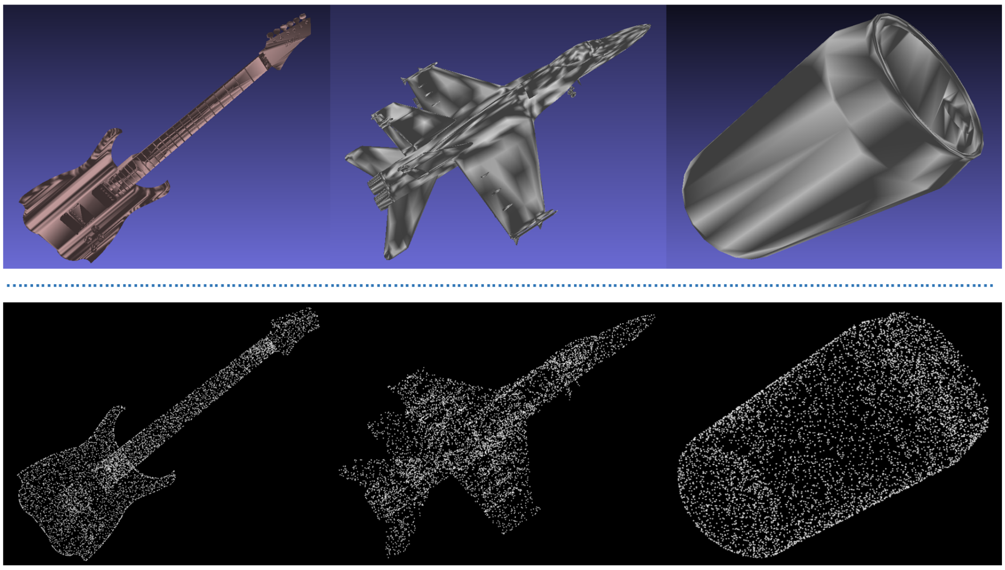


Figure 4.2: Sample meshes to point clouds. The top row shows the 3D CAD meshes from *ShapeNet* and the bottom is the obtained point clouds.

the performance both qualitatively and quantitatively. Following the literature [TSSF12][OKA11a][SKR$^+$15], we use the following error metrics for quantitative evaluation:

1. Mean joint error for all joints for each frame and average across all test frames in millimeter (mm), which measures the average Euclidean distance error for all joints across the whole test set and is defined as:

$$Error = \frac{\sum_{j=1}^{N} \sum_{i=1}^{21} |joint_i - joint_i^{gt}|/21}{N}, \tag{4.1}$$

where N is the total number of frames, $joint_i$ denotes the 3D coordinate of the i-th joint for curren frame and $joint_i^{gt}$ is the corresponding ground truth.

2. Mean finger tip error. Similarly to mean joint error, since some datasets contain only finger tip annotations, the mean joint error for 3D finger tip positions is used to evaluate these test datasets.

3. Correct frame proportion, which is the ratio of frames r_f that have all joints within a certain distance to ground truth annotation defined as:

$$r_f = \frac{N_f}{N}, \tag{4.2}$$

where N is the total number of frames, $N_f = g(\epsilon)$ is the number of frames whose joints are all within euclidean distance of ϵ to the ground truth.

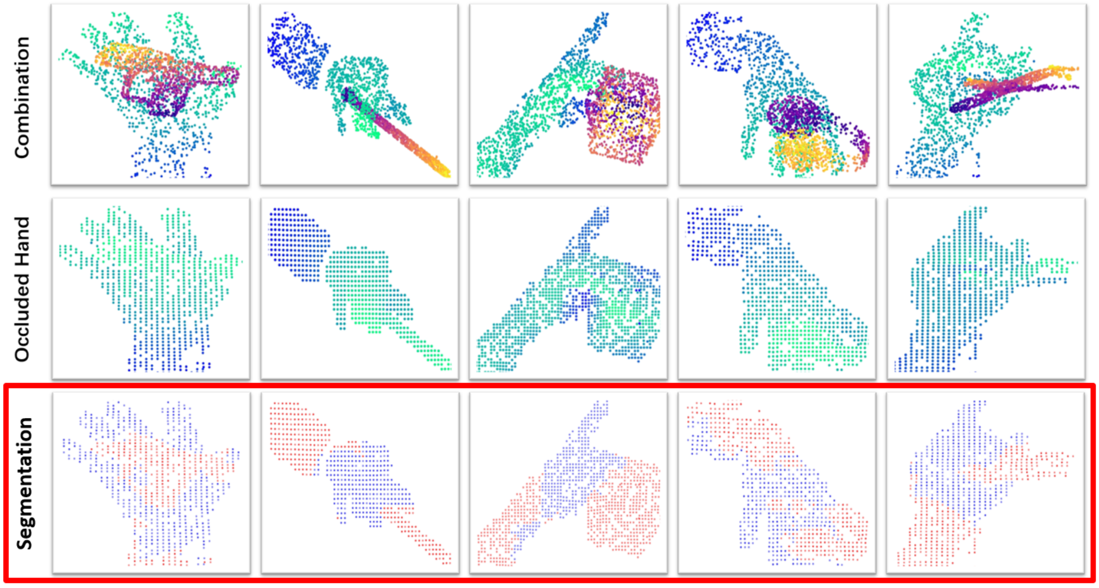


Figure 4.3: Segmentation results based on importance term. The hand-object point cloud is indicated with different color. Blue: important hand part for pose estimation, Red: less important part for pose estimation.

We evaluate the performance of our method only qualitatively on real data for the trained model on *Hands2017Challenge*, because it contains no annotated samples for hand-object interaction cases.

For the *SynthHands* dataset, two standard metrics are used for evaluation. The first one is the mean joint error in millimeter (mm) that measures the average Euclidean distance error for all joints across the whole test set. The second metric is correct frame proportion that indicates the percentage of frames that have all joint errors within a certain threshold compared to the ground-truth. The correct frame proportion metric is challenging, since a single joint violation will cause an incorrect frame.

For the *EgoDexter* dataset with only finger tip annotations, we use finger tip error for evaluation, which is the mean joint error for 3D finger tip positions.

4.4 Segmentation using Importance Term

In our Residual PEL based encoder, an importance term $\mathbf{G} \in \mathbb{R}^{N\times256}$ is computed with respect to the feature term $\mathbf{F_2} \in \mathbb{R}^{N\times256}$. Each element of $\mathbf{G}$ indicates the weight for corresponding feature value in $\mathbf{F_2}$ and provides vital information of the importance of current feature value. For the input point cloud $\mathbf{P}^a \in \mathbb{R}^{N\times3}$, we can get the importance vector $\mathbf{g} \in \mathbb{R}^N$ for all points by the Equation 4.3. By comparing this importance vector $\mathbf{g}$ with a certain threshold, the hand-object point cloud can be segmented into different parts.

$$\mathbf{g} = \sum_{i=1}^{256} \mathbf{G}_{ni}, \tag{4.3}$$

Based on the *Hands2017Challenge* dataset, some segmentation results for the augmented point clouds are shown in Fig. 4.3. The significant part of the hand is painted with blue. The points that have less contribution to pose estimation are indicated with red, such as the arm part and the object part. This result shows that our method is able to recognize insignificant object and arm parts of the occluded hand point cloud by applying smaller weights on those points.

4.5 Comparison of Reconstruction Decoders

We have tried two different 3D shape reconstruction methods in this work, which are the FoldingNet based decoder and the AtlasNet based decoder, respectively. In this section, we perform an experiment to compare the performance of the methods. We train the networks on *SynthHands*. Specifically, the training dataset composes of 75% clean hand samples and 25% interacting hand samples.

The qualitative results of both reconstruction decoders are shown in Fig. 4.4. As we can see from the illustration, more detailed structures of the hands are reconstructed by the FoldingNet based decoder compared to the AtlasNet based decoder. Intuitively, the 3D pose accuracy of the model using FoldingNet reconstruction decoder is much better. The detailed quantitative comparison can be found in Table 4.1.

Table 4.1: Quantitative comparison of FoldingNet and AtlasNet based decoders on *SynthHands*.

Decoders	**Error on Test Dataset (mm)**	
	clean hand	interacting hand
FoldingNet	**9.63**	**14.16**
AtlasNet	22.72	29.41

On clean hand test set, the mean joint error of FoldingNet is 9.63 mm compared to 22.72 mm of AtlasNet. On interacting test set, the mean joint error of AtlasNet is 15.25 mm more compared to FoldingNet.

We find that the pose accuracy of AtlasNet is much worse than FoldingNet, although it uses three more 5-layer perceptrons. Compared to AtlasNet, FoldingNet shows its outperformance with fewer parameters. The possible reason for this is that AtlasNet achieves state-of-the-art performance only with sufficient patches at the cost of more

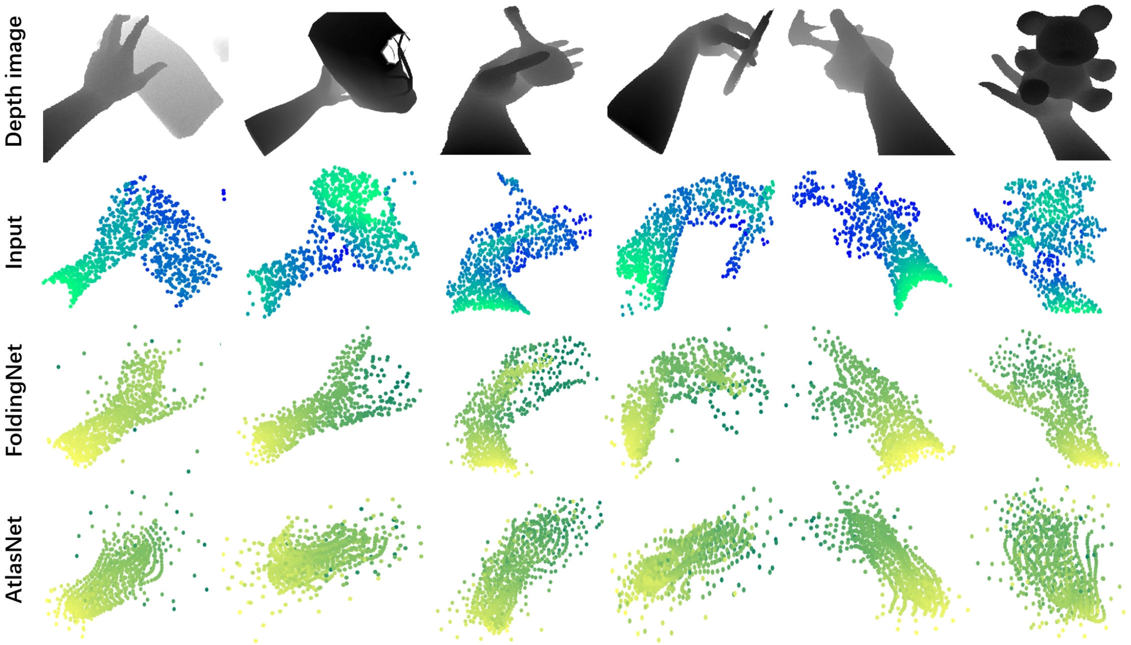


Figure 4.4: Qualitative comparison of FoldingNet and AtlasNet based decoders on *SynthHands*. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

parameters.

In our work, extremely high quality of clean hand reconstruction is not strictly required. Predicting the hand pose with the help of the auxiliary reconstruction Decoder 1 is our main objective. Considering the clean hand reconstruction quality and the complexity of our network, we choose FoldingNet based decoder as the clean hand reconstruction decoder in our framework and next experiments.

4.6 Comparison to state-of-the-art Method

To compare the performance of our method with state-of-the-art methods, we use the benchmark dataset *EgoDexter*.

Since the *EgoDexter* dataset is only annotated on 3D finger tip positions, we use the finger tip error to compare the performance of our method with the kinematic pose tracking method proposed by Mueller et al. [MMS+17]. We follow the same training dataset in their work, where all samples in *SynthHands* are used. As shown in Fig. 4.5, our method outperforms the state-of-the-art method on the test sequences, achieving the average error of 28.70 mm. Compared to 32.6 mm by Mueller et al., our method reduces the average error by 3.9 mm.

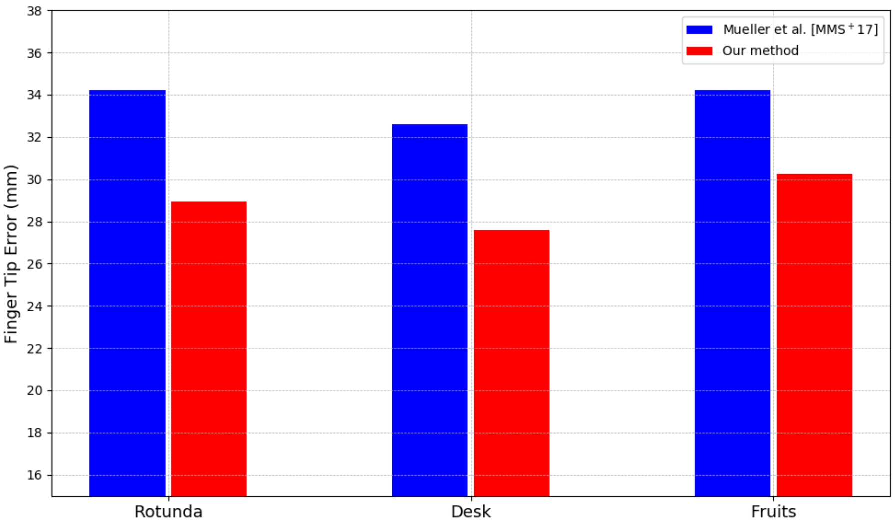


Figure 4.5: Comparison to state-of-the-art method on *EgoDexter* benchmark.

Note that the objects in *EgoDexter* are different from the objects in *SynthHands* training data. It shows the generalization ability of our method to unknown objects.

4.7 Ablation Studies

In this section, we perform two ablation experiments to investigate the effects of the augmentation component and the reconstruction component in our method.

Ablation Study 1

In the **first** ablation experiment, we mix different proportions of interacting hand samples to training set to compare the performance of different trained models. Then we use the optimal mixing proportion for the next experiments.

Using the training samples from *SynthHands*, we set 4 different training datasets with varying percentages of hand-object interaction samples:

- Dataset A: 100% clean hand samples.
- Dataset B: 75% clean + 25% interacting hand samples.
- Dataset C: 50% clean + 50% interacting hand samples.
- Dataset D: 25% clean + 75% interacting hand samples.

Note that the interacting hand samples are not augmented during training time. In other words, we only perform data augmentation on clean hand samples.

The detailed comparison of mean joint errors on our both test sets can be found in Table 4.2. We can already obtain a reasonably good result on 100% clean hand Dataset A. Even if using only augmented hand samples from clean hand without any interacting hand samples, the error on interacting test set is 19.13 mm, which indicates the effectiveness of the augmentation strategy.

Table 4.2: Comparison of different training methods on SynthHands.

Training Dataset	**Error on Test Dataset (mm)**	
	clean hand	interacting hand
A	9.67	19.13
B	**9.63**	**14.16**
C	10.69	14.35
D	12.52	15.99

Furthermore, the best performance is achieved with training Dataset B, which contains 25% interacting hand samples. Compared to Dataset A, the mean joint error is decreased from 19.13 mm to 14.16 mm on interacting hand test set by mixing only a small proportion of real interacting hand samples in the training dataset.

However, with the increasing proportion of interacting hand for training, the results become slightly worse, even on the interacting test set. The possible reason for this is that the decrease of clean hand proportion leads to less data augmentation, which means less random objects are seen for the training process, resulting in less generalizability on the unseen objects in the test set. Furthermore, for the interacting training samples, hand reconstruction is not performed since there is no available clean hand ground-truth for reconstruction, this leads to insufficient training of the reconstruction decoder and in turn influences the quality of the latent space.

This experiment shows that, in practice, we can utilize large clean hand dataset and mix a small proportion of interacting hand samples, which are expensive to annotate, to achieve robust performance.

Ablation Study 2

In the **second** experiment, for ablation study, we set the following baselines to show the effects of the data augmentation and points reconstruction approaches:

- Baseline 1. From the proposed method, we remove the random data augmentation process for the input clean hand point clouds.
- Baseline 2. From the proposed method, we remove the Decoder 1 in our framework, which is used to reconstruct clean hand point clouds for augmented input clean hands.

Table 4.3: Comparison with baselines on SynthHands.

Model	Error on Test Dataset (mm)	
	clean hand	interacting hand
Our method	**9.63**	**14.16**
Baseline 1	15.44	20.78
Baseline 2	19.60	23.46

Both baselines are trained using Dataset B. As seen in Fig. 4.6, our method outperforms the two baselines on both clean hand test set and interacting hand test set. Table 4.3 shows that the results of baselines are worse even on clean hand test set. The possible reason for this is that the latent representation in baselines is implicitly correlated to the mixture of clean hands and interacting hands rather than clean hands alone in our Augmented Autoencoder based framework. By this result, we demonstrate the significant effects of the augmentation component and the reconstruction component in our method.

4.8 Qualitative Results

Based on the *Hands2017Challenge* dataset, the reconstructed clean hand point clouds corresponding to the augmented occluded hands are illustrated in Fig. 4.7.

For the *SynthHands* dataset, the qualitative comparison of our method with two baselines is shown in Fig. 4.8 on the interacting test set.

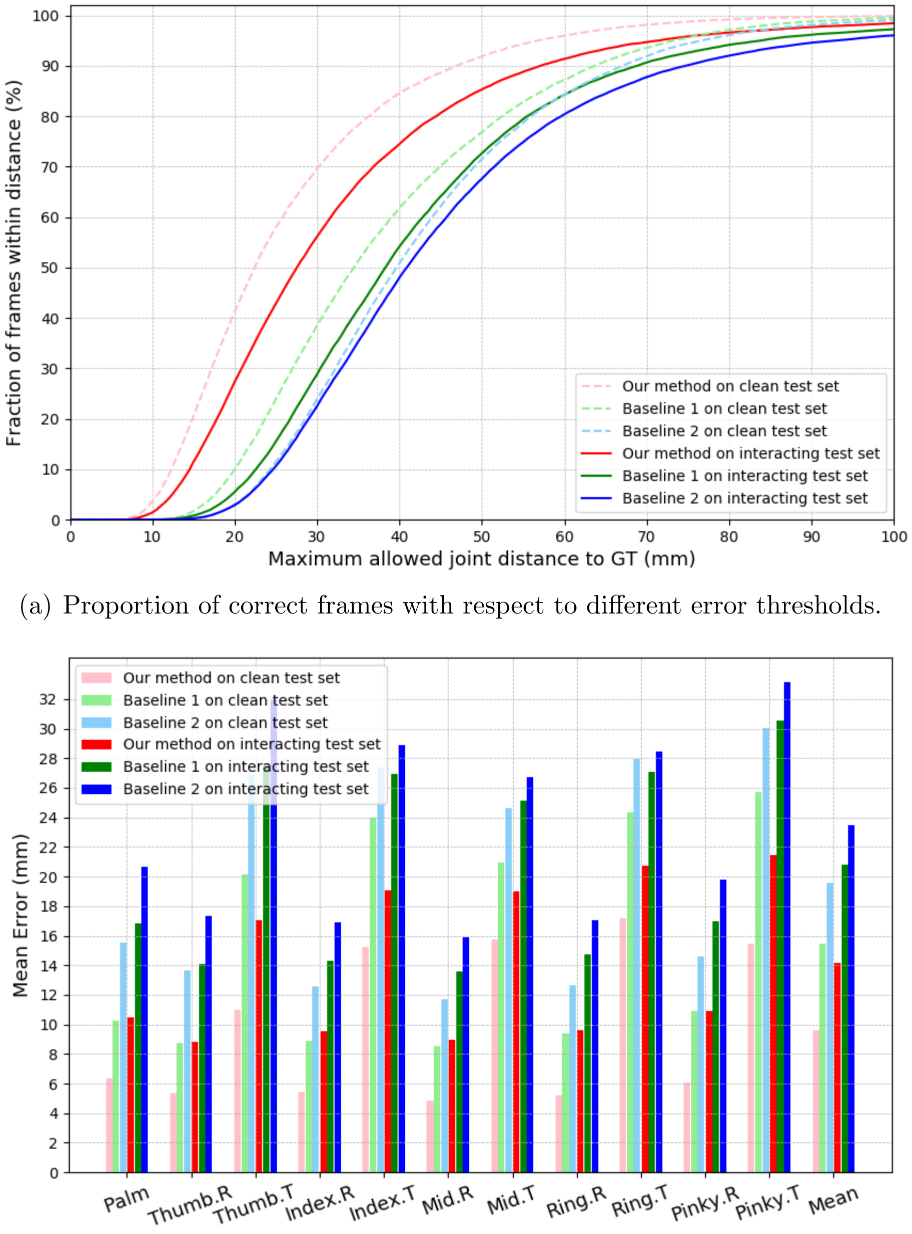
For the *Hands2017Challenge* dataset, as the training set and test set contain only clean hands, we train our model without mixing any interacting hands. Furthermore, we just give a qualitative result on the trained model with this dataset for evaluation. Fig. 4.9 shows qualitative results on real data, where the hand interacts with different objects, such as ball, bucket, phone, paper box. Although the model is trained only with clean hand data on the *Hands2017Challenge* dataset, the results shows good performance.

Note that high quality point cloud reconstruction is not strictly required in our method. Fig. 4.9 shows that occluded objects are roughly removed after reconstruction, indicating the importance of Decoder 1 for the formation of the latent space of the clean hand.

4.9 Runtime

Our method runs on a PC with an AMD FX-4300 Quad-Core CPU and an Nvidia GeForce GTX1060 6GB GPU. For the testing stage, the runtime of our method is

25.4 ms per frame with $N = 900$ points as input, which is able to run at real-time speed. The runtime can be further reduced only with a small performance loss when less points are used.



(a) Proportion of correct frames with respect to different error thresholds.

(b) Mean errors of different joints.

Figure 4.6: Comparison to baselines on *SynthHandsTest*.

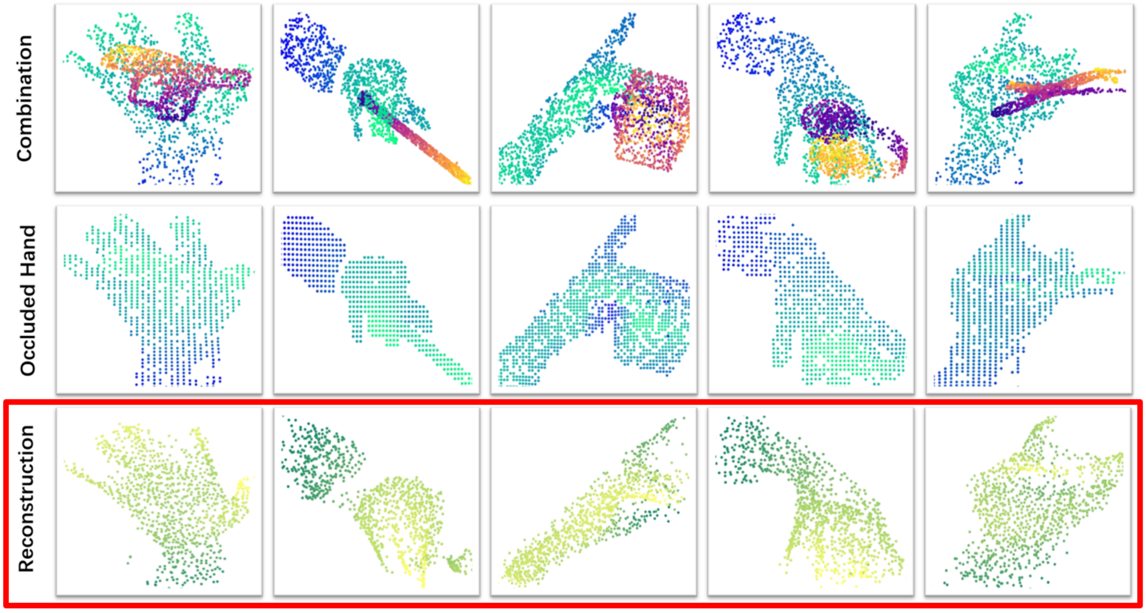


Figure 4.7: Reconstruction results corresponding the augmented occluded hands on *Hands2017Challenge*. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

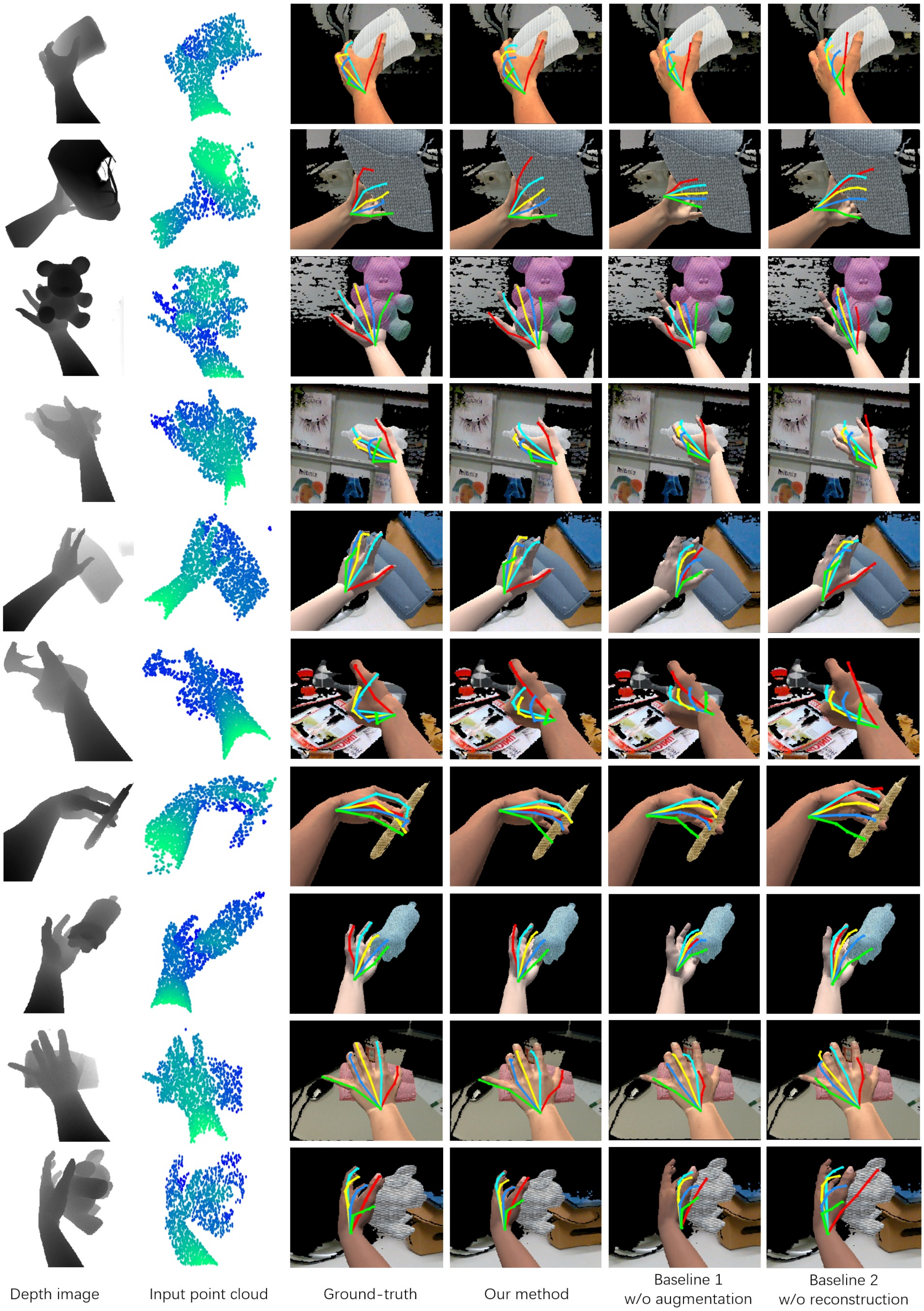


Figure 4.8: Qualitative results compared with baselines on *SynthHands*. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

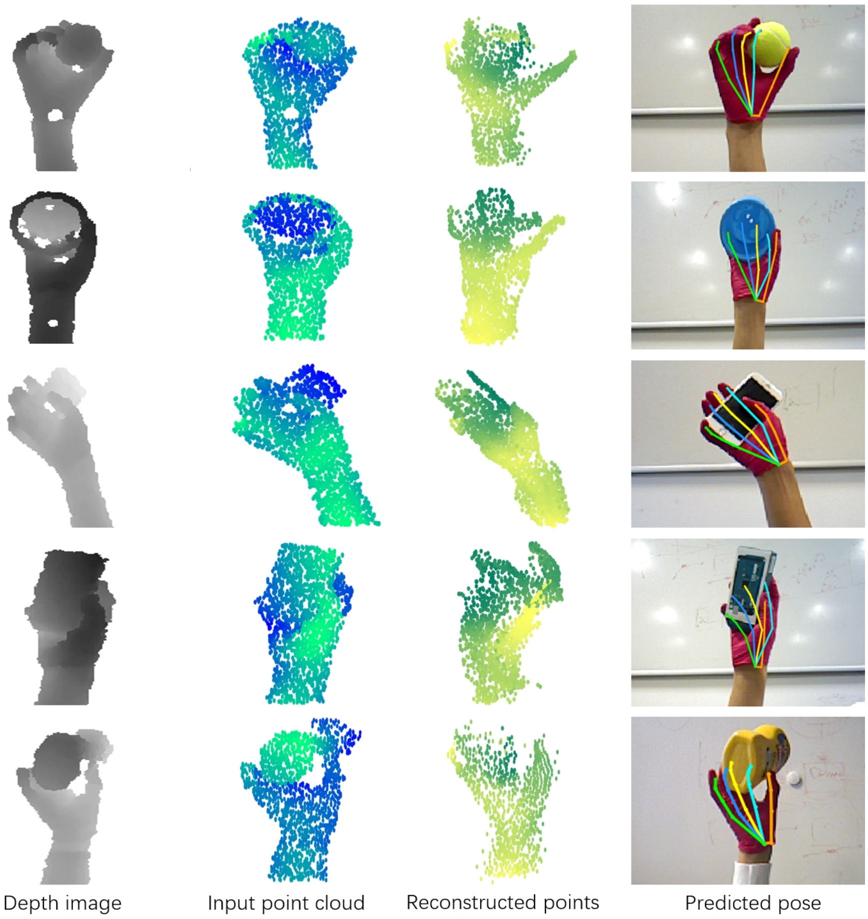


Figure 4.9: Qualitative results on real data. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

Chapter 5

Future Work

Although our method achieves the state-of-the-art performance, there exist some disadvantages in the deep network model, which deserve to be improved by future work.

5.1 Disadvantages of Our Method

As introduced in this paper, the most important two components in our method are the data augmentation strategy, which simulates hand-object interaction cases utilizing existing clean hand datasets, and the auxiliary decoder, which reconstructs clean hand point clouds. However, both coarse components limit the performance of our method to some extent and can be further optimized.

For the data augmentation process, we simply superimpose an object on a clean hand sample after random rotation, scaling and translation, where the physical constraints of hand-object interaction are not utilized. Specifically, the prior knowledge that contacts occur at the surface between the hand and object when grasping an object is not accounted. In other words, interpenetration happens in our coarse augmentation method. Moreover, the surfaces of the hand and object are occasionally not in contact. These cases violate the constrains of hand-object interaction in physical world and result in poor augmentation quality.

The clean hand reconstruction decoder plays an important role in our Augmented Autoencoder based method, however, as shown in Fig. 4.9, the occluded objects are only roughly removed from the input and the quality of reconstructed hand point clouds is not ideal as expected. For example, the delicate fingers are sometimes not clearly reconstructed and there are some joint angle violations with strange hand shape. Although high quality reconstruction of the clean hand point cloud is not the main objective in our method, the coarse reconstruction decreases the accuracy of the predicted pose.

5.2 Optimization Methods

Addressing the disadvantages stated above, we propose the following optimization approaches.

Offline augmentation to obtain realistic data

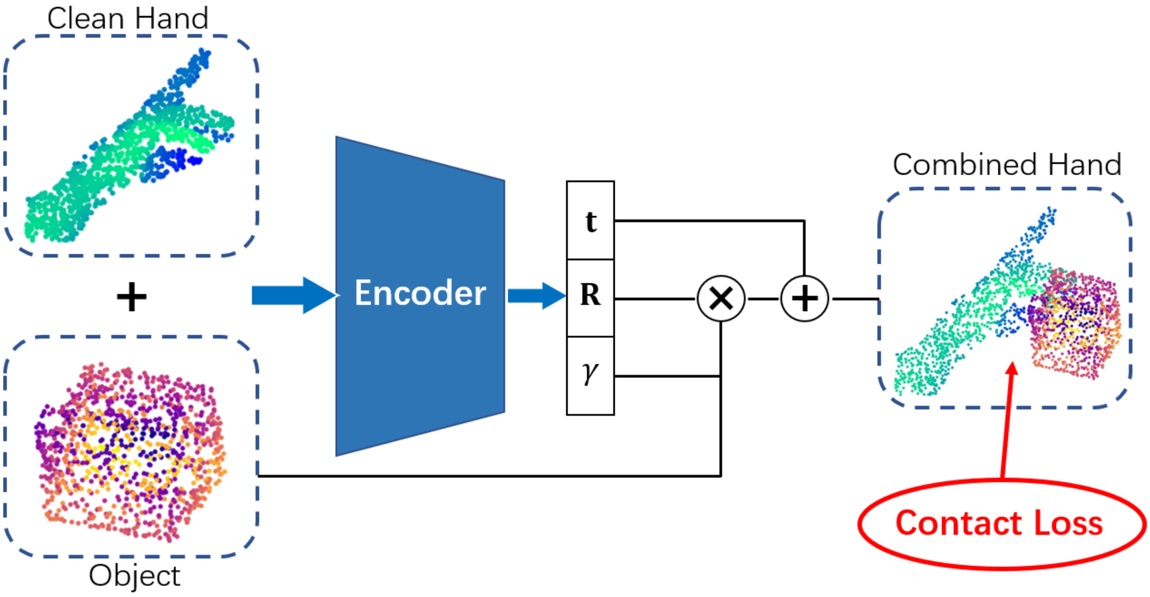


Figure 5.1: Offline augmentation using contact loss to acquire realistic data. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

For the violation of physical constraints for hand-object interaction cases, we introduce an offline augmentation approach with contact loss [HVT+19], which is illustrated in Fig. 5.1. The new augmentation approach uses both the clean hand sample and the object to regress a set of basic transformation parameters for the object, which are the scaling (γ), rotation ($\mathbf{R}$) and translation ($\mathbf{t}$) parameters. Similarly, after these basic transformations, the combined hand point cloud is obtained, which is then followed by rendering and sampling. The encoder for this offline augmentation pipeline is trained using the contact loss $\mathcal{L}_{Contact}(\gamma, \mathbf{R}, \mathbf{t})$, which is defined as the weighted sum of a repulsion term $\mathcal{L}_R$ and an attraction term $\mathcal{L}_A$:

$$\mathcal{L}_{Contact}(\gamma, \mathbf{R}, \mathbf{t}) = (1 - \lambda)\mathcal{L}_A + \lambda\mathcal{L}_R, \tag{5.1}$$

and the optimization goal is:

$$\min_{\gamma, \mathbf{R}, \mathbf{t}} \mathcal{L}_{Contact}(\gamma, \mathbf{R}, \mathbf{t}) \tag{5.2}$$

where λ is the weighting coefficient, e.g. $\lambda = 1$ means only the repulsion term is active. During training, a balance point for λ is supposed to be found to achieve satisfactory physical quality of augmentation.

During training stage, the repulsion loss $\mathcal{L}_R$ will penalize hand and object interpenetration. To detect interpenetration, we count the number of hand points that are inside the object. The attraction loss $\mathcal{L}_A$ will penalize the cases in which the key surface regions of a hand are not in contact with the surface of an object. The key surface regions are defined as the areas on the hand which are frequently involved in contacts. The attraction term $\mathcal{L}_A$ penalizes distances from each of these key regions to the object surface.

Model-based Reconstruction Decoder

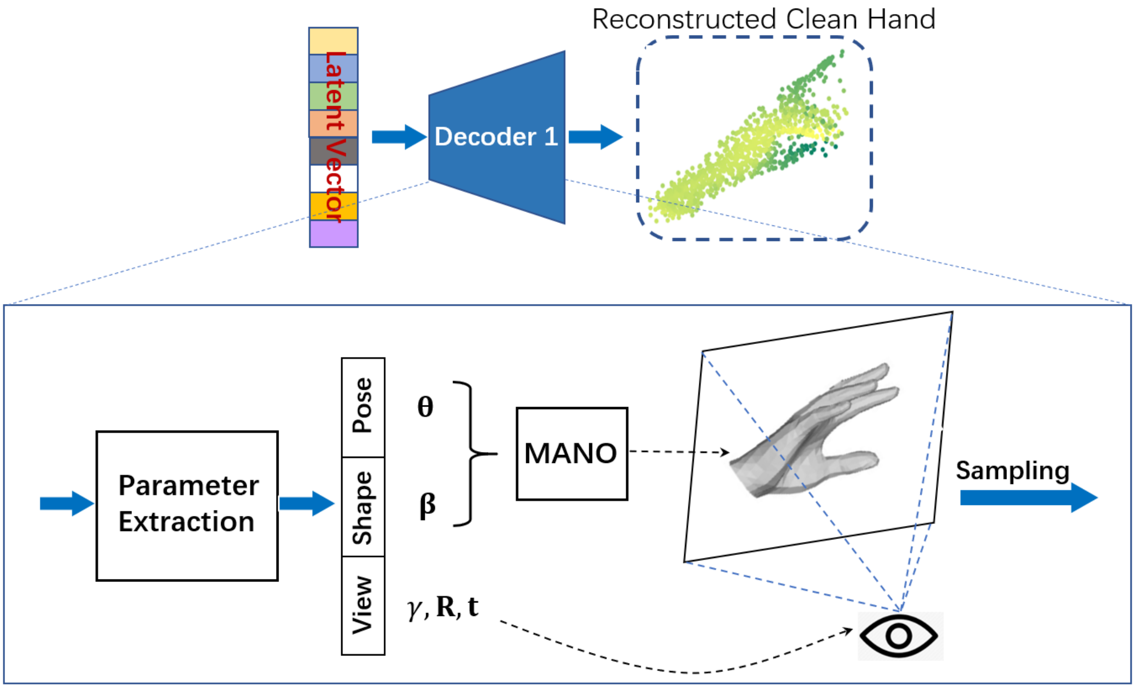


Figure 5.2: Hand model (MANO) based clean hand reconstruction decoder. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

To improve the quality of reconstructed clean hand point clouds, one possible method is to use more folding operations in the FoldingNet based decoder. For AtlasNet based decoder, more patches can be invoked to construct more delicate hand structure. However, this kind of methods are computationally expensive at the cost of more parameters. In this section, we propose a hand model-based reconstruction decoder to replace the pure deep learning based decoder network. The structure of the proposed decoder network is illustrated in Fig. 5.2.

We can use the MANO hand model [RTB17] which is based on the SMPL model for human bodies [LMR$^+$15]. It is an articulate deformation hand model that maps the pose parameter ($\boldsymbol{\theta}$) and the shape parameter ($\boldsymbol{\beta}$) to a mesh with a differentiable mapping function. While the pose parameter captures the the angles between the hand joints, the shape parameter controls the person-specific deformations of the hand.

As shown in Fig. 5.2, a parameter extraction deep network processes the input latent vector and generates the pose parameter $\boldsymbol{\theta}$, the shape parameter $\boldsymbol{\beta}$ and also a set of view parameters γ, $\mathbf{R}$, $\mathbf{t}$. The hand parameters are fed to the mesh deformation hand model to generate a 3D mesh. Through a perspective camera model controlled by the view parameters, we sample the hand mesh to final points, which is our reconstructed clean hand point cloud.

This proposed model-based method removes the cases of joint angle violations and also strange hand shapes in the current reconstructed hands. The quality of the latent space can be further optimized, which in turn improves the hand pose accuracy.

Chapter 6

Conclusion

In this paper, we propose a novel deep learning framework using Augmented Autoencoder to handle hand pose estimation tasks for hand-object interaction cases. Our method consumes 3D hand point cloud and predicts accurate 3D hand pose. The proposed augmentation process and auxiliary clean hand reconstruction decoder implicitly force the latent representation of the pose only to be correlated to clean hand and the reconstructed clean hand despite the object occlusion in hand-object interaction cases. Furthermore, the proposed hand pose estimation training strategy is able to utilize existing clean hand datasets to tackle hand-object interaction cases. Quantitative and qualitative evaluation results show that our framework is capable of achieving low joint errors on both clean hand input ($\sim 9\ mm$) and interacting hand input ($\sim 14\ mm$) on the *SynthHandsTest*. Our method demonstrates state-of-the-art performance for hand pose estimation with objects.

The proposed data augmentation strategy is able to utilize existing large clean hand datasets to simulate hand-object interaction cases in manipulation tasks. This strategy provides a good solution for the lack of existing available hand-object interaction datasets. Furthermore, since unlimited types of objects can be used for augmentation, the proposed model is more generalizable on unknown objects. By ablation experiments, we show that, in practice, a small proportion of real interacting hand samples can be mixed into the training dataset to acquire much robuster performance.

In our method, the proposed auxiliary decoder is of great importance to achieve high hand pose estimation accuracy for hand-object interaction cases. The encoder takes occluded hand point cloud as input and outputs the latent representation of the hand pose. The novel auxiliary decoder reconstructs corresponding clean hand points simultaneously, which makes the latent representation only be correlated to clean hands alone instead of the mixture of clean hands and interacting hands. Although high quality of clean hand reconstruction is not strictly required, this 3D point cloud reconstruction operation implicitly improves the quality of the latent space a lot. In turn, the hand pose accuracy for hand-object interaction cases is improved.

Appendix A

DVD

List of Figures

1.1 A previous hand pose estimation work [LL19] fails in hand-object interaction tasks. 5

1.2 The raw data are captured from a RGB-D camera. We use only the depth image to acquire the input cloud. The RGB image is used for visualization. For the output, besides the predicted pose, a clean hand is simultaneously reconstructed. (Brightness in point cloud indicates depth, i.e. darker denotes further.) 6

3.1 Overview of our method. The input of our network is an occluded hand point cloud, which is obtained by a random augmentation process from a clean hand point cloud. The encoder encodes the input hand to a latent vector. The obtained latent vector is then used to reconstruct clean hand point cloud by the auxiliary Decoder 1 and predict 3D hand pose by Decoder 2. There are three losses in our VAE based framework, which are the KL loss, reconstruction loss and pose loss. (Brightness in point cloud indicates depth, i.e. darker denotes further.) 14

3.2 View normalization for hand point clouds. Red skeletons indicate ground-truth pose, green points indicate observed points of the camera. a) Due to different view directions, different observations lead to the same hand pose, thus the resulted training samples will contain one-to-many mappings. b) With view normalization, the different observations will also have different pose labels, thus the input-output pairs have the one-to-one mapping relation. [LL18] 15

3.3 Combination results. (Brightness in point cloud indicates depth, i.e. darker denotes further.) 16

3.4 Project combination results to depth images. (Brightness indicates depth, i.e. darker denotes further.) 18

3.5 Project depth images to final occluded hand point clouds. Brightness indicates depth value, namely, darker point denotes that the point is more distanced to camera, and vice versa. 18

3.6 Training process for the AAE. a) reconstruction target batch of uniformly sampled SO(3) object views; b) geometric and color augmented input; c) reconstruction results. [SMD+18] 19
3.7 A training-time VAE with reparameterization trick. Dash line box shows loss layers, $\boldsymbol{\mu}(\mathbf{x})$ and $\boldsymbol{\Sigma}(\mathbf{x})$ denote the mean and the variance of the latent variable $\mathbf{z}$, ϕ and θ signify model parameters. 20
3.8 Residual PEL based encoder network. K denotes the number of dimensions of latent variables, dash line box describes the detailed structure for one of the residual blocks. 21
3.9 FoldingNet based decoder network for points reconstruction. K denotes the number of dimensions of latent variables. 23
3.10 The shape generation approach for AtlasNet. [GFK+18] 24
3.11 AtlasNet based decoder network for points reconstruction. K denotes the number of dimensions of latent variables. 25

4.1 Joint annotation [YYGHK17] 30
4.2 Sample meshes to point clouds. The top row shows the 3D CAD meshes from *ShapeNet* and the bottom is the obtained point clouds. . 32
4.3 Segmentation results based on importance term. The hand-object point cloud is indicated with different color. Blue: important hand part for pose estimation, Red: less important part for pose estimation. 33
4.4 Qualitative comparison of FoldingNet and AtlasNet based decoders on *SynthHands*. (Brightness in point cloud indicates depth, i.e. darker denotes further.) 35
4.5 Comparison to state-of-the-art method on *EgoDexter* benchmark. . . 36
4.6 Comparison to baselines on *SynthHandsTest*. 40
4.7 Reconstruction results corresponding the augmented occluded hands on *Hands2017Challenge*. (Brightness in point cloud indicates depth, i.e. darker denotes further.) 41
4.8 Qualitative results compared with baselines on *SynthHands*. (Brightness in point cloud indicates depth, i.e. darker denotes further.) . . . 42
4.9 Qualitative results on real data. (Brightness in point cloud indicates depth, i.e. darker denotes further.) 43

5.1 Offline augmentation using contact loss to acquire realistic data. (Brightness in point cloud indicates depth, i.e. darker denotes further.) . . . 46
5.2 Hand model (MANO) based clean hand reconstruction decoder. (Brightness in point cloud indicates depth, i.e. darker denotes further.) . . . 47

Acronyms and Notations

CD Chamfer Distance

EMD Earth Mover's Distance

VAE Variational Autoencoder

KL Loss Kullback-Leibler Loss

PEL Permutation Equivariant Layer

AAE Augmented Autoencoder

Bibliography

[ABC$^+$16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[ACVB09] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[BBT19] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 3d hand shape and pose from images in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10843–10852, 2019.

[BTG$^+$12] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. Motion capture of hands in action using discriminative salient points. In *European Conference on Computer Vision*, pages 640–653. Springer, 2012.

[CFG$^+$15] Angel Xuan Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

[CHYCR17] Chiho Choi, Sang Ho Yoon, Chin-Ning Chen, and Karthik Ramani. Robust hand pose estimation during the interaction with an unknown object. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3123–3132, 2017.

[CKS$^+$16] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

[CXG$^+$16] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[FSG17] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[GCWY18] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. Hand pointnet: 3d hand pose estimation using point sets. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8417–8426, 2018.

[GFK$^+$18] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-m\^ ach\'e approach to learning 3d surface generation. *arXiv preprint arXiv:1802.05384*, 2018.

[GHYBK18] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 409–419, 2018.

[GLYT16] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3593–3601, 2016.

[GLYT17] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1991–2000, 2017.

[GWF$^+$19] Yafei Gao, Yida Wang, Pietro Falco, Nassir Navab, and Federico Tombari. Variational object-aware 3d hand pose from a single rgb image. *IEEE Robotics and Automation Letters*, 2019.

[HSKMVG09] Henning Hamer, Konrad Schindler, Esther Koller-Meier, and Luc Van Gool. Tracking a hand manipulating an object. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1475–1482. IEEE, 2009.

[HVT+19] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11807–11816, 2019.

[HYW+15] Chaoqun Hong, Jun Yu, Jian Wan, Dacheng Tao, and Meng Wang. Multimodal deep autoencoder for human pose recovery. *IEEE Transactions on Image Processing*, 24(12):5659–5670, 2015.

[JNC+15] Youngkyoon Jang, Seung-Tak Noh, Hyung Jin Chang, Tae-Kyun Kim, and Woontack Woo. 3d finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint. *IEEE Transactions on Visualization and Computer Graphics*, 21(4):501–510, 2015.

[KA13] Nikolaos Kyriazis and Antonis Argyros. Physically plausible 3d scene tracking: The single actor hypothesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9–16, 2013.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[KW14] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[LL18] Shile Li and Dongheui Lee. Point-to-pose voting based hand pose estimation using residual permutation equivariant layer. *CoRR*, abs/1812.02050, 2018.

[LL19] Shile Li and Dongheui Lee. Point-to-pose voting based hand pose estimation using residual permutation equivariant layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11927–11936, 2019.

[lLLY19] Yang linlin, Shile Li, Dongheui Lee, and Angela Yao. Aligning latent spaces for 3d hand pose estimation. In *Proceedings of the International Conference on Computer Vision*, 2019.

[LMR+15] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):248, 2015.

[MBS+18] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian

Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018.

[MEC$^+$17] Meysam Madadi, Sergio Escalera, Alex Carruesco, Carlos Andujar, Xavier Baró, and Jordi Gonzàlez. Occlusion aware hand pose recovery from sequences of depth images. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 230–237. IEEE, 2017.

[MMS$^+$17] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1284–1293, 2017.

[N$^+$11] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[OKA11a] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BmVC*, volume 1, page 3, 2011.

[OKA11b] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *2011 International Conference on Computer Vision*, pages 2088–2095. IEEE, 2011.

[OWL15] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015.

[PCBC13] Thammathip Piumsomboon, Adrian Clark, Mark Billinghurst, and Andy Cockburn. User-defined gestures for augmented reality. In *IFIP Conference on Human-Computer Interaction*, pages 282–299. Springer, 2013.

[POA18] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. Using a single rgb frame for real time 3d hand pose estimation in the wild. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 436–445. IEEE, 2018.

[QYSG17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[RTB17] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (TOG)*, 36(6):245, 2017.

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[SKR$^+$15] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3633–3642. ACM, 2015.

[SMD$^+$18] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018.

[SMZ$^+$16] Srinath Sridhar, Franziska Mueller, Michael Zollhöfer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *European Conference on Computer Vision*, pages 294–310. Springer, 2016.

[SSPH18] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. Cross-modal deep variational hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–98, 2018.

[SUHR17] Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6040–6049, 2017.

[TA18] Aggeliki Tsoli and Antonis A Argyros. Joint 3d tracking of a deformable object in interaction with a hand. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 484–500, 2018.

[TBP19] Bugra Tekin, Federica Bogo, and Marc Pollefeys. H+ o: Unified egocentric recognition of 3d hand-object poses and interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4511–4520, 2019.

[TBS$^+$16] Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision*, 118(2):172–193, 2016.

[TSSF12] Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 103–110. IEEE, 2012.

[VLL$^+$10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.

[WPVGY17] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 680–689, 2017.

[WPVGY18] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Dense 3d regression for hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2018.

[YFST18] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2018.

[YK18] Qi Ye and Tae-Kyun Kim. Occlusion-aware hand pose estimation using hierarchical mixture density network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–817, 2018.

[YY19] Linlin Yang and Angela Yao. Disentangling latent hands for image synthesis and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9877–9886, 2019.

[YYGHK17] Shanxin Yuan, Qi Ye, Guillermo Garcia-Hernando, and Tae-Kyun Kim. The 2017 hands in the million challenge on 3d hand pose estimation. *arXiv preprint arXiv:1707.02237*, 2017.

[YYS$^+$17] Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhant Jain, and Tae-Kyun Kim. Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4866–4874, 2017.

[ZB17] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4903–4911, 2017.

[ZBYX19] Hao Zhang, Zi-Hao Bo, Jun-Hai Yong, and Feng Xu. Interactionfusion: real-time reconstruction of hand poses and deformable objects in hand-object interactions. *ACM Transactions on Graphics (TOG)*, 38(4):48, 2019.

[ZRG$^+$07] Loredana Zollo, Stefano Roccella, Eugenio Guglielmelli, M Chiara Carrozza, and Paolo Dario. Biomechatronic design and control of an anthropomorphic artificial hand for prosthetic and robotic applications. *IEEE/ASME Transactions On Mechatronics*, 12(4):418–429, 2007.

License