Real-time 3D hand pose estimation from depth images

Ge, Lihao

2018

Ge, L. (2018). Real-time 3D hand pose estimation from depth images. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/87652>

<https://doi.org/10.32657/10220/46781>

REAL-TIME 3D HAND POSE ESTIMATION FROM DEPTH IMAGES

LIUHAO GE

Interdisciplinary Graduate School

Institute for Media Innovation

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2018

To my family for their encouragement and support.

Acknowledgments

First and foremost, I would like to express my sincere gratitude and appreciation to my supervisors: Professor Junsong Yuan, Professor Jianfei Cai, Professor Daniel Thalmann, Professor Nadia Magnenat Thalmann, and Professor Kai-Kuang Ma. Thank you for your guidance, encouragement and support during my PhD study. Your constructive and insightful advices help me to become a capable researcher. Your encouragement and strong support help me overcome difficulties encountered in my research and life. The time working with you is really a great pleasure and precious memory.

My thanks also go to Dr. Hui Liang, Dr. Zhou Ren, Yujun Cai, Junwu Weng, Tan Yu, Dr. Zhiwen Fang, Dr. Chunluan Zhou, Dr. Jiong Yang, Dr. Kang Dang, Dr. Chaoqun Weng, Dr. Zhigang Tu, Dr. Mengyuan Liu, and Dr. Junlin Hu. It is really joyful to have you share your insights on research and life. I will always remember the time we work and play together. Especially, I would like to thank Dr. Hui Liang for his guidance and suggestions when I was a junior PhD student, and Dr. Zhou Ren for his support and help during the internship in the last year of my PhD study.

I would also like to acknowledge the fund and Ph.D. grant from the Institute for Media Innovation (IMI) and the Interdisciplinary Graduate School (IGS), Nanyang Technological University, Singapore. I am honored to be a student of IGS and study in IMI. I appreciate IMI for providing excellent environment and computing

facilities for my research.

I am greatly indebted to my dear grandparents and parents. Your unconditional love and support have always been my continuous source of strength and inspiration.

Contents

Acknowledgments	i
Summary	viii
List of Figures	x
List of Tables	xiv
1 Introduction	1
1.1 Problem Definition	1
1.2 Challenges	3
1.3 Thesis Organization and Contributions	4
2 Related Work	6
2.1 Marker-based Hand Pose Estimation	6
2.2 Markerless Hand Pose Estimation	7
2.2.1 Single RGB-based Hand Pose Estimation	8
2.2.2 Multi-camera-based Hand Pose Estimation	9
2.2.3 Depth Camera-based Hand Pose Estimation	10
2.2.4 Other Sensors	10

2.3	Hand Pose Estimation from Depth Images	10
2.3.1	Model-driven Methods	11
2.3.2	Data-driven Methods	12
2.3.3	Hybrid Methods	13
2.4	CNN-based Methods for Hand Pose Estimation from Depth Images	14
2.5	3D Hand Pose Datasets	15
3	Multi-view CNNs-based 3D Hand Pose Estimation	17
3.1	Overview	17
3.2	Multi-view Representation	20
3.2.1	Projection with Oriented Bounding Box	21
3.2.2	Projection with Axis-Aligned Bounding Box	22
3.3	Multi-View Convolutional Neural Networks	24
3.3.1	Network Architectures	24
3.3.2	Loss Function for Heat-map Regression	24
3.3.3	3D Data Augmentation on Projected Images	25
3.4	Multi-view Fusion Algorithm	27
3.4.1	Problem Formulation	27
3.4.2	View Selection	30
3.4.3	Solution to the Optimization Problem	31
3.5	Experiments	34
3.5.1	Implementation Details	34
3.5.2	Self-Comparison	35
3.5.3	Comparison with State-of-the-Art	40

3.5.4	Cross-Dataset Experiment	42
3.5.5	Qualitative Results	43
3.5.6	Runtime	45
3.6	Summary	46
4	3D CNN-based 3D Hand Pose Estimation	47
4.1	Overview	47
4.2	Volumetric Representations	50
4.2.1	Occupancy Grid	51
4.2.2	TSDF Volumes	51
4.3	3D Convolutional Neural Networks	54
4.3.1	Network Architectures	54
4.3.2	Loss Function for 3D Hand Pose Regression	55
4.3.3	3D Data Augmentation	56
4.4	Hand Surface Completion	58
4.5	Visualizing 3D CNN	59
4.6	Experiments	61
4.6.1	Implementation Details	61
4.6.2	Self-comparison	62
4.6.3	Comparison with State-of-the-art	66
4.6.4	Cross-dataset Experiment	70
4.6.5	Qualitative Results	71
4.6.6	Runtime and Model Size	72
4.7	Applications	73

4.8	Summary	74
5	Point Set-based Holistic Regression for Hand Pose Estimation	76
5.1	Overview	76
5.2	PointNet Revisited	78
5.3	Point Cloud Normalization	79
5.4	Hand Pose Regression Network	82
5.5	Fingertip Refinement Network	84
5.6	Experiments	85
5.6.1	Implementation Details	86
5.6.2	Self-comparisons	87
5.6.3	Comparisons with State-of-the-art Methods	89
5.6.4	Qualitative Results	90
5.6.5	Runtime and Model Size	91
5.7	Summary	91
6	Point Set-based Point-wise Regression for Hand Pose Estimation	92
6.1	Overview	92
6.2	Point Set-based Representation of 3D Hand Pose	95
6.3	Network Architecture of Point-wise Regression PointNet	97
6.4	Hand Pose Inference	99
6.4.1	Hand Pose Inference from Point-wise Estimation	99
6.4.2	Post-processing	100
6.5	Experiments	102
6.5.1	Implementation Details	102

6.5.2	Self-comparisons	102
6.5.3	Comparisons with State-of-the-arts	104
6.5.4	Qualitative Results	105
6.5.5	Runtime and Model Size	106
6.6	Summary	107
7	Conclusions and Future Works	108
7.1	Conclusions	108
7.2	Future Works	109
	Author's Publications	111
	Bibliography	113

Summary

Accurate and real-time 3D hand pose estimation is one of the core technologies for human computer interaction in virtual reality and augmented reality applications, since this technology provides a natural way for users to interact with virtual environments and virtual objects. Despite the previous works in this field, it is still challenging to achieve efficient and robust hand pose estimation performance because of large variations in hand pose, high dimensionality of hand motion, severe self-occlusion and self-similarity of fingers.

This thesis focuses on the problem of 3D hand pose estimation from single depth images. To achieve accurate results and real-time performance of 3D hand pose estimation, four different methods are proposed, which are multi-view CNNs-based method, 3D CNN-based method, point set-based holistic regression method, and point set-based point-wise regression method. Overall, this thesis has made the following major contributions:

- A multi-view CNNs-based 3D hand pose estimation method is proposed in this thesis. To better exploit 3D information in the depth image, the point cloud generated from the query depth image is projected onto multiple views of two projection settings. Multi-view CNNs are trained to learn the mapping from projected images to heat-maps which reflect probability distributions of joints on each view. These multi-view heat-maps are fused to estimate optimal 3D hand pose with learned pose priors. Unreliable information in multi-view heat-maps is suppressed using a view selection method.

- A 3D CNNs-based 3D hand pose estimation method is proposed in this thesis. The 3D CNN takes a 3D volumetric representation of the hand depth image as input and extracts 3D features from the volumetric input. To further improve the estimation accuracy, the complete hand surface is used as intermediate supervision for learning 3D hand pose from depth images.
- A point set-based holistic regression method for 3D hand pose estimation is proposed in this thesis. The proposed Hand PointNet directly processes the 3D point cloud for holistic hand pose regression. Taking the normalized point cloud as the input, the hand pose regression network is able to capture complex hand structures and accurately regress a low dimensional representation of the 3D hand pose. To further improve the accuracy of fingertips, a fingertip refinement network is proposed.
- A point set-based point-wise regression method for 3D hand pose estimation is proposed in this thesis. The proposed Point-to-Point Regression PointNet directly takes the 3D point cloud as input and outputs point-wise estimations, *i.e.*, heat-maps and unit vector fields on the point cloud, representing the closeness and direction from each point in the point cloud to the hand joint. The point-wise estimations are used to infer 3D joint locations with weighted fusion.

Experimental results on public hand pose datasets have shown that all these proposed methods are able to achieve superior accuracy performance and run in real-time on GPU. Some hand pose estimation applications in virtual reality environments are developed in this thesis.

List of Figures

1.1	Overview of 3D hand pose estimation from depth images.	2
1.2	Hand skeleton strcuture.	2
1.3	Organization of this thesis.	4
2.1	Marker-based hand pose estimation methods.	7
2.2	Markerless hand pose estimation methods.	9
2.3	Hand models used in model-driven methods.	12
2.4	Different hand joint definations in different datasets.	16
3.1	Overview of our proposed multi-view CNNs-based approach for 3D hand pose estimation.	18
3.2	Two problems of single view CNN-based method.	19
3.3	Illustration of multi-view representation.	21
3.4	Examples of projected images on six views.	22
3.5	Network architectures for multi-view CNNs	25
3.6	An example of 3D data augmentation.	26
3.7	Illustration of multi-view fusion.	29
3.8	Self-comparison of different methods on MSRA2015 dataset [1] using OBB's three views.	35
3.9	A case of ambiguous estimation.	37

3.10	Self-comparison of different heat-map resolutions on NYU [2] hand pose dataset.	38
3.11	Evaluation of combining OBB and AABB's six views, view selection method, 3D data augmentation and different network architectures.	39
3.12	Comparison with state-of-the-art approaches on MSRA2015 dataset [1].	40
3.13	Comparison with state-of-the-art approaches on NYU dataset [2].	41
3.14	Qualitative results on MSRA2015 dataset [1] and NYU dataset [2].	44
3.15	Qualitative results for testing in real scenarios on different subjects with different hand sizes and hand poses.	44
4.1	Different schemes for 3D hand pose estimation.	48
4.2	Visualization of different volumetric representations.	52
4.3	3D plain and dense network architectures.	53
4.4	An example of 3D data augmentation.	57
4.5	The framework for hand surface completion and 3D hand pose estimation.	59
4.6	Visualization of patterns learned in a fully trained 3D CNN model.	59
4.7	Self-comparison of different volume resolutions and different volume types with/without data augmentation on MSRA2015 dataset [1].	62
4.8	Self-comparison of 2D/3D CNNs with different network architectures for 3D hand pose estimation on three datasets.	64
4.9	Examples of hand surface completion with our method on NYU hand pose dataset [2].	66
4.10	Comparison of 3D CNN-based method with state-of-the-art methods on MSRA2015 dataset [1].	67

4.11	Comparison of 3D CNN-based method with state-of-the-art methods on NYU dataset [2].	68
4.12	Comparison with state-of-the-art methods of 3D CNN-based method on ICVL dataset [3].	69
4.13	Qualitative results for MSRA2015, NYU and ICVL datasets.	71
4.14	Qualitative results of 3D CNN-based method testing in real scenarios using Intel RealSense SR300 Depth Camera.	72
4.15	Applications of our proposed 3D CNN-based hand pose estimation method.	74
5.1	Overview of our proposed Hand PointNet-based method with holistic regression for 3D hand pose estimation in single depth images. .	77
5.2	Basic architecture of PointNet.	79
5.3	Examples of OBB-based point cloud normalization.	80
5.4	Architecture of the proposed point set-based holistic hand pose regression network.	83
5.5	Visualization of the point sensitivity in different regions to different filters at three set abstraction levels.	84
5.6	Illustration of fingertip refinement.	85
5.7	Self-comparison of different methods on NYU dataset [2].	87
5.8	The impact of fingertip refinement on fingertips' mean error distances and the overall mean error distance.	88
5.9	Comparison of point set-based holistic regression method with state-of-the-art methods (curve of good frame percentages).	89
5.10	Comparison of point set-based holistic regression method with state-of-the-art methods (mean error distances).	90

5.11	Qualitative results of point set-based holistic regression method. . .	90
6.1	Overview of our proposed point-wise regression method for 3D hand pose estimation from single depth images.	93
6.2	An illustration of the ground truth of the point cloud based repre- sentation for 3D hand pose.	96
6.3	An illustration of a single Point-to-Point regression PointNet module.	98
6.4	An illustration of the two-stacked hierarchical PointNet architecture with intermediate supervision.	99
6.5	Post-processing of point set-based point-wise regression	101
6.6	Self-comparison of different methods on NYU dataset [2].	103
6.7	Comparison of point set-based point-wise regression method with state-of-the-art methods (curve of good frame percentages).	106
6.8	Comparison of point set-based point-wise regression method with state-of-the-art methods (mean error distances).	106
6.9	Qualitative results of point set-based point-wise regression method.	107

List of Tables

3.1	Cross-dataset experiment for multi-view CNNs-based method. . . .	43
4.1	Impact of Hand Surface Completion on 3D Hand Pose Estimation using 3D ResNet and 3D DenseNet.	66
4.2	Cross-dataset experiment for 3D CNN-based method.	70
6.1	The impacts of the number of candidate estimations M and weight- ed average on the overall mean error distance on NYU dataset [2]. .	104

Chapter 1

Introduction

Recent years have witnessed a rapid growth in virtual reality (VR) and augmented reality (AR) applications. Mechanical devices, such as keyboards, mice, and joysticks, are cumbersome and unsuitable for immersive interactions in VR/AR applications. Can users directly use their bare hands to naturally interact with the virtual objects in the VR/AR environment? Thanks to the development of camera techniques and computer vision, vision-based hand pose estimation has become an important and promising technology for human-computer interaction.

1.1 Problem Definition

With the emergence of commercial depth cameras in the past ten years, *e.g.*, Microsoft Kinect, Intel RealSense, Primesense Carmine, many research works have focused on 3D hand pose estimation from depth images [2, 4–12]. Compared with traditional cameras that capture intensity or color in the scene, depth cameras are robust in low-light situation, can capture 3D spatial information of the object and are invariant to color and texture. These properties make depth camera suitable for robust 3D hand pose estimation.

In this thesis, we investigate the problem of 3D hand pose estimation from depth images. As presented in Figure 1.1, the input is a depth image capturing the hand gesture. Human hand is segmented from the input depth image for hand pose estimation. The output is a set of 3D hand joint locations representing the

Figure 1.1: Overview of 3D hand pose estimation from depth images. The input is a depth image; human hand is segmented from the depth image for hand pose estimation; the output is a set of 3D hand joint locations representing the 3D hand pose. The estimated 3D hand pose can be applied in various applications. The left two snapshots of applications are from our self-built systems. The right two snapshots of applications are cropped from [13] and [9], respectively.

3D hand pose. The estimated 3D hand pose can be used for various applications, such as virtual object manipulation, virtual 3D drawing, virtual keyboard, etc.

The skeleton structure of hand is shown in Figure 1.2. As shown in this figure, the thumb finger has 5 degrees of freedom (DOF), including 1 DOF for interphalangeal (IP) joint, 2 DOF for metacarpophalangeal (MCP) joint and 2 DOF for trapeziometacarpal (TM) joint. Each of the other four fingers has 4 DOF, including 1 DOF for distal interphalangeal (DIP) joint, 1 DOF for proximal interphalangeal (PIP) joint, and 2 DOF for MCP joint. The wrist joint has 6 DOF for global translation and rotation. Thus, the hand skeleton model has 27 DOF in

Figure 1.2: Hand skeleton structure defined in [14]. The thumb finger has 5 DOF; each of the other four fingers has 4 DOF; the wrist joint has 6 DOF.

total. This thesis aims at estimating the full DOF 3D hand pose from single depth images.

1.2 Challenges

Although 3D hand pose estimation from depth images has aroused a lot of research attention in recent years, it is still challenging to achieve efficient and robust hand pose estimation performance.

Firstly, estimating 3D hand pose from depth images is a high-dimensional and non-linear regression problem. As illustrated in Section 1.1, the hand skeleton model has 27 DOF, including 6 DOF for global hand motion and 21 DOF for local finger motion. Due to the high-dimensional space of the hand pose parameters, it is difficult to find the optimal hand pose in a large parameter space when using the generative hand pose estimation method. When using data-driven method for hand pose estimation, it is hard to cover all the variations of the 3D hand pose in the training data, and the training could be difficult since the mapping between the depth image and the 3D hand pose is highly non-linear.

Secondly, hand pose in single depth images often suffers from severe self-occlusion problem. Because of large variations in local and global hand motion, some parts of the hand may be occluded by other parts of the hand in a single depth image. Thus, 3D hand pose estimation from single depth images may be ambiguous. In the data-driven method, the ambiguity of the occluded hand pose may cause the training process more difficult. Some pose constraints and priors could be applied to infer the most plausible hand pose.

Thirdly, the depth images captured from range sensors are usually noisy and may have data missing. Estimating 3D hand pose from noisy and incomplete depth image is difficult. In addition, the discriminative model trained on synthetic data with on noise or data missing may not perform well on the real data. Generating more realistic synthetic data may be a solution. But it is difficult to simulate all the kinds of noise and data missing in the synthetic training data.

Figure 1.3: Organization of this thesis.

1.3 Thesis Organization and Contributions

Figure 1.3 illustrates the organization of this thesis. Chapter 1 gives an introduction to this thesis and presents the problems and challenges to address in this thesis. Chapter 2 presents a literature review on hand pose estimation methods and related datasets. Our proposed methods for 3D hand pose estimation from depth images are presented in Chapter 3, Chapter 4, Chapter 5, and Chapter 6, including multi-view CNNs-based method, 3D CNN-based method, point set-based holistic regression method, and point set-based point-wise regression method. The conclusions and future works are discussed in Chapter 7.

More specifically, we list the major contributions of the works in Chapter 3, Chapter 4, Chapter 5, and Chapter 6 as follows:

Chapter 3. In this chapter, a multi-view CNNs-based approach for robust 3D hand pose estimation is proposed. To better leverage 3D information in the depth image, a multi-view representation of the depth image is proposed. Multi-view CNNs are trained to map projected images to heat-maps. A multi-view fusion algorithm is proposed to obtain the optimal estimation of 3D hand joint locations. This work has been published in the IEEE Conference on Computer Vision and Pattern Recognition, 2016 [15]. The extended version has been published in the IEEE Transaction on Image Processing [16].

Chapter 4. In this chapter, a 3D CNN-based hand pose estimation method for 3D hand pose estimation is proposed. The hand depth image is encoded as a 3D volumetric representation, and is fed into a 3D CNN. To tackle the self-occlusion problem, the complete hand surface is used as intermediate supervision for learning more accurate 3D hand pose. This work has been published in the IEEE Conference on Computer Vision and Pattern Recognition, 2017 [17]. The extended version has been accepted by the IEEE Transaction on Pattern Analysis and Machine Intelligence [18].

Chapter 5. In this chapter, a point set-based holistic regression method for 3D hand pose estimation is proposed. The 3D point cloud extracted from the depth image is directly taken as network input to holistically regress 3D hand joint locations. To handle variations of hand global orientations, the hand point cloud is normalized with more consistent global orientations. This work has been published in the IEEE Conference on Computer Vision and Pattern Recognition, 2018 [19].

Chapter 6. In this chapter, a point set-based point-wise regression method for 3D hand pose estimation is proposed. The proposed network directly takes the 3D point cloud as input and generates heat-maps as well as unit vector fields on the input point cloud. A post-processing method is proposed to infer 3D hand joint locations from the estimated heat-maps and unit vector fields. This work has been accepted by the European Conference on Computer Vision, 2018 [20].

Chapter 2

Related Work

Since directly using hand as an interface for human-computer interaction is very attractive but challenging, a lot of research works have focused on hand pose estimation in the past thirty years [2, 12, 14, 18, 21–27]. In this chapter, we present a literature review on hand pose estimation methods. We first classify hand pose estimation methods into marker-based methods and markerless methods. Then, we focus on discussing the methods of hand pose estimation from depth images, especially the methods using convolutional neural networks (CNNs). We also review public datasets for 3D hand pose estimation from depth images.

2.1 Marker-based Hand Pose Estimation

Optical marker-based motion capture has been widely applied in commercial systems for video games, films and television [28]. Some research works have adopted the marker-based method in hand pose estimation [13, 28–37], as shown in Figure 2.1.

Theobalt *et al.* [30] track the hand motion by placing 18 markers with four different colors on the position of hand joints on the glove. Four cameras and one stroboscope are used to capture and reconstruct the hand motion. Wang and Popović [31] use a single camera to track the hand wearing a multi-colored glove. The glove is imprinted with 20 large patches using 10 unique colors. The 3D hand pose is inferred by using the nearest-neighbor method.

Figure 2.1: Marker-based hand pose estimation methods. (a) Dense marker setup for the hand motion capture (mocap) system [28]. (b) Sparse marker setup for online hand motion capture [33]. (c) Marker-based method for capturing the hand grasping motion [35]. (d) Online marker-based hand tracking system [13].

Instead of using dense marker set [28], as shown in Figure 2.1a, for hand pose estimation, some work aims at reducing the number of markers. As shown in Figure 2.1b, Alexanderson *et al.* [33] propose to use sparse marker set with three to ten markers per hand for hand motion capture. This method adopts several hypotheses and soft decisions for hand pose reconstruction. Schröder *et al.* [35] propose a method for determining the sparse marker layout by analyzing hand motion. In Figure 2.1c, they demonstrate their method is able to capture the grasping motion. Han *et al.* [13] propose a robust hand tracking system, as shown in Figure 2.1d. They use convolutional neural networks to regress the position of marker points, and use inverse kinematics to reconstruct the hand pose from the detected markers.

Although marker-based hand pose estimation methods have achieved some good results, it is inconvenient for users to place markers on their hand or wear a glove with markers. It is promising to perform markerless hand pose estimation.

2.2 Markerless Hand Pose Estimation

In this section, we classify the markerless hand pose estimation method based on the sensors. We review hand pose estimation methods using monocular camera, multiple cameras, single depth camera and other sensors, respectively.

2.2.1 Single RGB-based Hand Pose Estimation

The most common sensor for capturing hand motion is the monocular camera. But it is very challenging to recover 3D hand pose from single RGB images. Due to the limitation of sensors, early works on hand pose estimation mainly use monocular cameras [38–44].

Model-based methods use a 3D hand model and minimize the difference between the projection of the hand model and the observed image. Stenger *et al.* [38, 39] propose to adapt the 2D profiles of the hand model to detected edges of the hand image, and combine hierarchical detection with Bayesian filtering for hand tracking. Wu *et al.* [41] propose a divide-and-conquer method in order to decouple the hand global pose and finger articulations. A cardboard hand model is adopted in this method to match the image observation using Iterative Closed Point (ICP) algorithm. Gorce *et al.* [44] propose a detailed generative hand model that can model 3D hand pose, surface texture and illuminant. Gradient with respect to hand pose and lighting is derived in order to minimize the difference between the synthesized image and the observed image.

Discriminative methods learn a discriminative model from training data. Wu and Huang [40] propose a Discriminant Expectation-Maximization (D-EM) approach which is able to utilize small labelled training set and large unlabelled training set in an effective way. Athitsos and Sclaroff [42] formulate the task of hand pose estimation as a database retrieval problem. They propose two clutter-tolerant indexing approaches to find the best matched hand pose in the dataset for the given image. But traditional discriminative RGB-based methods suffer from complex background, self-occlusion, and large pose variation, thus have limited generalization ability.

Recently, with the great success of deep neural networks, some researchers have applied convolutional neural networks (CNNs) on this challenging task [45–50]. RGB-based hand pose estimation using CNNs is a promising direction in the future.

Figure 2.2: Markerless hand pose estimation methods. (a) Single RGB image-based hand pose estimation method [45]. (b) Multi-camera-based hand pose estimation method [51]. (c) Single depth image-based hand pose estimation method [2]. (d) Millimeter-wave radar-based hand gesture sensing [52].

2.2.2 Multi-camera-based Hand Pose Estimation

Due to the difficulty of inferring 3D hand pose from single RGB images, some research works use a multi-camera setup for capturing 3D hand motion [51, 53–56], as shown in Figure 2.2b.

Rehg and Kanade [53] present a DigitEye system which can recover a 27 DOF hand model from two cameras. Kinematic models and feature models are used to describe the spatial link and image appearance of the hand. Ballan *et al.* [54] capture the motion of two hands with an object in a setup of eight synchronized cameras. They consider edge, optical flow, salient points and collisions in the objective function. Sridhar *et al.* [51] use five synchronized RGB cameras and one depth camera to capture hand motion. They combine a discriminative method with a generative method for 3D hand pose estimation. Joo *et al.* [56] present a unified 3D human model for capturing human motion including hand gestures. They capture multi-view images in the Panoptic Studio [57, 58] which is a massively multi-view system. The full-body human model is fitted to 3D points reconstructed from the multi-view images.

Although multi-camera-based approaches can achieve accurate estimation results, the complex multi-camera setup makes this kind of method difficult to be applied in commercial VR/AR systems.

2.2.3 Depth Camera-based Hand Pose Estimation

With the availability of mid-range and short-range depth cameras, such as Intel RealSense, SoftKinetic and Primesense Carmine, a lot of research works have been focused on 3D hand pose estimation from depth images [2, 4–10], as shown in Figure 2.2c. Methods for hand pose estimation from depth images can be categorized into model-driven approaches, data-driven approaches and hybrid approaches. More detailed review on these depth image-based approaches are presented in Section 2.3.

2.2.4 Other Sensors

Recently, some methods propose to use other sensors to capture the 3D hand motion. Leap Motion [59] adopts two infrared cameras and three infrared LEDs for 3D hand tracking. The infrared cameras can output grayscale stereo images of the near-infrared light spectrum [59]. Wang *et al.* [60] and Lien *et al.* [52] propose to use Google’s Soli sensor for capturing hand motion. As shown in Figure 2.2d, Google’s Soli sensor is a millimeter-wave radar that is able to capture rich properties of a moving object, including distance, velocity, shape, etc.

2.3 Hand Pose Estimation from Depth Images

With the success of real-time human body pose estimation using depth cameras [61–63], accurate real-time 3D hand pose estimation from depth images has aroused a lot of research attention in the past ten years [2–4, 7–12, 27]. Methods for hand pose estimation from depth images can be categorized into model-driven approaches, data-driven approaches and hybrid approaches [64].

2.3.1 Model-driven Methods

Model-driven approaches fit an explicit deformable hand model to depth images by minimizing a hand-crafted cost function. The commonly used optimization methods are Particle Swarm Optimization (PSO) [27], Iterative Closest Point (ICP) [65] and their combination [66]. Many hand models have been proposed, as shown in Figure 2.3. Oikonomidis *et al.* [67] propose a polygonal mesh hand model using geometric primitives. Qian *et al.* [66] approximate the 3D hand model using 48 spheres. Sridhar *et al.* [68] propose a Gaussian mixture model of hand which is also applied in [62, 69]. Melax *et al.* [70] propose to use a rigid-body representation of the hand model. Taylor *et al.* [9] create the hand model based on linear blend skinning and approximate Loop subdivision, which is also applied in [55, 71]. Tkach *et al.* [72] propose to use sphere-meshes as the 3D hand model. Romero *et al.* [73] propose a realistic hand model which can capture non-rigid shape of hand pose and is learned from 3D scans of hands. Joo *et al.* [56] use a rigged hand mesh as the hand model for total capture of human.

However, there are some shortcomings for the model-driven methods. For instance, some model-driven methods usually need to explicitly define the anatomical size and hand motion constraints of the hand to match to the input depth image. Also, due to the high dimensional of hand pose parameters, they can be sensitive to initialization for the iterative model-fitting procedure to converge to the optimal pose.

Figure 2.3: Hand models used in model-driven methods. (a) Hand model using geometric primitives [67]. (b) Hand model using 48 spheres [66]. (c) Hand model using Gaussian mixture [68]. (d) Hand model used in [70]. (e) Hand model using linear blend skinning and approximate Loop subdivision [9]. (f) Hand model using sphere-meshes [72]. (g) Hand MANO model proposed in [73]. (h) Hand model used in [56].

2.3.2 Data-driven Methods

In contrast to model-driven methods, the data-driven methods do not need the explicit specification of the hand size and motion constraints. Rather, such information is automatically encoded in the training data. Therefore, many recent methods are built upon such a scheme [1, 4, 5, 74–76].

Data-driven approaches learn a mapping from depth image to hand pose from training data. Some early works [77–79] focus on example-based method that searches the most similar images in a dataset to the input hand image, but cannot work well in high dimensional space. Inspired by the pioneering work [61] of human pose estimation, [1, 3, 4, 80–83] apply random forests and their variants as a discriminative model. In [4], the authors propose to use the random forest to directly regress for the hand joint angles from depth images, in which a set of spatial-voting pixels cast their votes for hand pose independently and their

votes are clustered into a set of candidates. The optimal one is determined by a verification stage with a hand model. A similar method is presented in [74], which further adopts transfer learning to make up for the inconsistency between synthesis and real-world data. As the estimations from random forest can be ambiguous for complex hand postures, pre-learned hand pose priors are sometimes utilized to better fuse independently predicted hand joint distributions [7, 84]. In [1], the cascaded pose regression algorithm [85] is adapted to the problem of hand pose estimation. Particularly, the authors propose to first predict the root joints of the hand skeleton, based on which the rest joints are updated. In this way the hand pose constraints can be well preserved during pose regression.

Limited by the hand-crafted features, data-driven methods based on random forests are difficult to outperform current CNN-based methods in hand pose estimation. We will present a detailed review of CNN-based methods in Section 2.4.

2.3.3 Hybrid Methods

Some other approaches combine the model-based fitting and the data-driven methods in order to take advantages of both methods [8, 9, 68, 75, 86, 87]. In [8], a multi-layered discriminative model, which is trained on synthetic training data, is first applied to infer hand poses. Then, the optimized hand pose is estimated by a model-fitting stage based on particle swarm optimization. In [68], a random decision forest is trained to detect hand parts. Then, a model-based generative tracking method is applied to estimate hand pose by combining hand part labels and Gaussian mixture representation of depth data into an objective function.

2.4 CNN-based Methods for Hand Pose Estimation from Depth Images

With the success of deep neural networks in various computer vision tasks and the emergence of large hand pose datasets [1–3, 11, 12], many of the recent 3D hand pose estimation methods are based on CNNs [2, 5, 15, 17, 88–93].

Tompson et al. [2] first propose to apply CNNs in 3D hand pose estimation. They use CNNs to generate heat-maps representing the 2D probability distributions of hand joints in the depth image, and recover 3D hand pose from estimated heat-maps and corresponding depth values using model-based inverse kinematics. Oberweger et al. [92, 94] instead directly regress 3D coordinates of hand joints or a lower dimensional embedding of 3D hand pose from depth images. They also propose a feedback loop network [5] to iteratively refine the 3D hand pose. Zhou et al. [95] propose to directly regress hand model parameters from depth images. Sinha et al. [96] extract activation features from CNNs to synchronize hand poses in nearest neighbors by using the matrix completion algorithm. Ye et al. [88] propose a spatial attention network with a hierarchical hybrid method for hand pose estimation. Guo et al. [90] propose a region ensemble network that directly regresses 3D hand pose from depth images. Chen et al. [93] improve [90] through iterative refinement. Wan et al. [97] propose a dense pixel-wise estimation method that applies an hourglass network to generate 2D and 3D heat-maps as well as 3D unit vector fields, from which the 3D hand joint locations can be inferred. Moon et al. [98] propose a voxel-to-voxel prediction method that estimates per-voxel probability in a 3D volume of each hand joint using a 3D CNN. Rad et al. [99] consider the problem of transfer learning between synthetic images and real images. They map the real image features to the feature space of synthetic images. This method can leverage the large synthetic hand pose dataset and limited real data with hand pose annotation.

Due to the difficulty of manually labeling 3D hand joint location ground truth on real depth images, some research works focus on leverage the large amount of unlabeled data. Wan et al. [89] propose to use deep generative models to estimate the shared latent space of the depth image and the hand pose parameters, which

is able to learn from unlabeled depth images. Baek et al. [100] propose to augment hand skeleton data and transfer them to real depth images. They exploit generative adversarial networks (GAN) to jointly train networks for hand pose estimation and hand depth image generation. Poier et al. [101] use multi-view images of the same hand pose to learn a latent representation of the hand pose. The network model can be trained with labeled and unlabeled data.

2.5 3D Hand Pose Datasets

Some hand pose datasets have been created in recent years, *i.e.*, NYU dataset [2], ICVL dataset [3], MSRA2014 dataset [66], MSRA2015 dataset [1], and Big Hand 2.2M dataset [11].

The NYU hand pose dataset [2] contains more than 72K frames for training and 8K frames for testing, which are captured by the PrimeSense™ 3D sensor. The ground truth of each frame contains 36 hand joints' 3D locations which are labeled using a model fitting method [27]. Previous works [2, 5] estimate a subset of $K = 14$ hand joints' 3D locations including three joints for thumb, two joints for index, middle, ring and little fingers, one joint for palm center and two joints for the wrist, as shown in Figure 2.4a. In our experiments in the following chapters, we will estimate the same subset of hand joints on this dataset.

The ICVL hand pose dataset [3] contains 12 training sequences having 22K frames and two testing sequences having 1.6K frames captured by the Intel's Creative Interactive Gesture Camera. This dataset additionally provides a training set in which the original training samples are in-plane rotated 14 times. The ground truth of each frame contains $K = 16$ hand joints' 3D locations including three joints for each finger and one joint for the palm center, as shown in Figure 2.4b. The ground truth of this dataset is labeled using the model-based method in [70] and is then manually refined.

The MSRA2014 hand pose dataset [66] contains six subjects' hand depth images captured by the Intel's Creative Interactive Gesture Camera. Each subject contains 400 frames. Thus, there are 2,400 frames in this dataset. The ground truth of each frame contains $K = 21$ hand joints' 3D locations including four joints for

Figure 2.4: Different hand joint definitions in different datasets. (a) 14 hand joints defined in NYU dataset [2]. (b) 16 hand joints defined in ICVL dataset [3]. (c) 21 hand joints defined in MSRA2014 [66], MSRA2015 [1], and Big Hand 2.2M [11] datasets.

each finger and one joint for the wrist, as shown in Figure 2.4c. The ground truth of this dataset is manually labeled.

The MSRA2015 hand pose dataset [1] contains nine subjects’ hand depth images captured by the Intel’s Creative Interactive Gesture Camera. Each subject performs 17 hand gestures, and each hand gesture contains about 500 frames. There are more than 76K frames in this dataset. Previous works [1, 5], eight subjects are used for training and the remaining subject is used for testing. This is repeated nine times for all subjects. In our experiments in the following chapters, we will perform the same experiments on this dataset. The definition of hand joints is the same as that in the MSRA2014 hand pose dataset [66]. The ground truth of this dataset is automatically annotated using the model-based method in [66] and is then manually corrected.

The Big Hand 2.2M hand pose dataset [11] contains ten subjects’ hand depth images captured by the Intel’s RealSense SR300 camera. This dataset contains 2.2M labeled frames with a large view point coverage. In order to obtain the ground truth of each frame, the subject is attached with six magnetic sensors at the back of palm. Inverse kinematics is used to solve the 21 hand joint locations from the six sensors’ outputs. However, since this dataset has not been released publically, we will not evaluate our methods on this dataset.

Chapter 3

Multi-view CNNs-based 3D Hand Pose Estimation

3.1 Overview

One of primary issues of the CNN-based approach for accurate 3D pose estimation is how to effectively utilize the depth image. If the neural network directly maps the 2D depth image to 3D joint locations, it will suffer from high learning complexity and low generalization ability, since the mapping is highly nonlinear [102–104]. To tackle this issue, Thompson et al. [2] proposed mapping input depth image to a set of heat-maps representing the 2D probability distributions of hand joints on the image plane and recovering 3D locations using the depth information with model fitting. Nonetheless, this approach cannot effectively exploit the 3D information in the depth image, since the estimated heat-maps only provide 2D information of hand joints projected on the image plane.

In this chapter, we propose a novel multi-view CNN-based 3D hand pose estimation method that can effectively utilize depth information to accurately infer 3D hand joint locations without model fitting, as depicted in Figure 3.1. In detail, human hand is first segmented from the input depth image; the point cloud generated from the hand depth image is projected onto multiple projection planes; the projected image is then fed into its corresponding network to regress a set of

Figure 3.1: Overview of our proposed multi-view CNNs-based approach for 3D hand pose estimation. We generate heat-maps for multiple views by projecting 3D points onto a number of projection planes. Multi-view CNNs are trained in parallel to map projected images on multiple views to their corresponding heat-maps, which are then fused together to estimate 3D hand joint locations.

heat-maps encoding the 2D probability distributions of hand joints on the corresponding projection plane. The 3D probability distribution of hand joint can be inferred from the combination of heat-maps on multiple views. By formulating multi-view fusion as the maximum a posteriori estimation with pre-learned hand pose constraints, we are able to obtain the optimal 3D hand pose and mitigate the ambiguity of estimations.

Compared with the single view CNN-based method [2], our proposed multi-view CNN-based method has the following advantages:

- The single view CNN-based method [2] directly takes the depth value at the inferred 2D hand joint position as the hand joint depth. As presented in Figure 3.2a, the error of depth estimation may be large, even if the inferred joint position just deviates a little from the true joint position on the image plane. On the contrary, our proposed method, generating and fusing heat-maps on multiple views, can estimate 3D locations more robustly.

Figure 3.2: Two problems of single view CNN-based method. (a) A case of hand pose estimation in single view. Blue points are true locations, and red points are estimated locations. Small 2D estimation error on the image plane may cause large depth estimation error. The tip of little finger is misestimated on the background and the tip of middle finger is misestimated on the palm. (b) A case of ambiguous estimation. We project the depth image onto x - y , y - z and z - x planes of a Cartesian coordinate system. Green dot indicates the true joint location in 2D projection plane, while red dot indicates the wrong joint location. Although the heat-map on x - y view contains two hotspots that are difficult to determine, from the heat-map on z - x view, it is obvious that the x value should be small with high confidence. Therefore, the left hotspot in the heat-map on x - y view should be true.

- In the case of ambiguous estimation, as presented in the x - y view of Figure 3.2b, the single view CNN cannot determine the true joint position among multiple hotspots in the heat-map. When adopting our proposed multi-view CNNs, the ambiguity can be eliminated by using heat-maps on other views, such as the case in Figure 3.2b.
- Different with the single view CNN-based method [2] that requires a pre-defined hand model to optimize estimated hand joint locations, our proposed method implicitly imposes hand pose constraints learned from training samples in the optimization problem, instead of manually defining hand model parameters.

The contributions of this chapter are summarized as follows:

- We have proposed a multi-view representation which projects the 3D point cloud onto six views of the oriented bounding box (OBB) and the axis-aligned bounding box (AABB) to better leverage 3D information in the depth image. Experimental results have shown that the combination can further boost the estimation performance.

- We have designed two networks by applying architectures of the residual network (ResNet) and the fully convolutional network (FCN) for estimating more accurate heat-maps.
- We have proposed a 3D data augmentation method for training CNNs of AABF's three views which are not rotation invariant, to make the multi-view CNNs more robust to various global hand orientations.
- We have proposed a view selection method to suppress unreliable information in heat-maps for multi-view fusion.
- We have conducted extensive self-comparison experiments and have compared with existing methods on both MSRA2015 dataset [1] and NYU dataset [2]. We have also evaluated our method in real scenarios using the SoftKinetic's DepthSense camera.

3.2 Multi-view Representation

The objective for multi-view representation is to generate a set of projected images $\{I_n\}_{n=1}^N$ on multiple views from the depth image of the segmented hand. As illustrated in Figure 3.1, the segmented hand in the depth image I_D is first converted to a set of 3D points in the camera coordinate system by using the depth camera's intrinsic parameters. To generate the multi-view representation, we project these 3D points onto N projection planes. In order to fully utilize the depth information in the 2.5D depth image, the projected images of one frame should provide complementary information as much as possible. Thus, orthogonal planes, on which the projected images can reflect different features of hand shapes from independent views, are good candidates for projection planes.

Figure 3.3: Illustration of multi-view representation. 3D hand points obtained from the segmented hand depth image are projected onto x - y , y - z and z - x planes of the OBB projection coordinate system and the AABB projection coordinate system, respectively. Color bars of 3D points and projected images are shown in this figure.

3.2.1 Projection with Oriented Bounding Box

We first project 3D points onto three orthogonal side planes of the oriented bounding box (OBB), as shown in Figure 3.3 (left). The OBB is generated by performing principal component analysis (PCA) on the set of 3D points, which is a tight fit around these 3D points in local space [105]. The origin of the OBB projection coordinate system is set at the center of the bounding box, and its x , y , z axes are respectively aligned with the 1st, 2nd and 3rd principal components. Since OBB is rotation invariant, projected images on OBB's side planes are also rotation invariant. Thus, this projection method using OBB is robust to variation in global hand orientations.

Figure 3.4: Examples of projected images on six views which are x - y , y - z , z - x views in the OBB projection coordinate system and x - y , y - z , z - x views in the AABB projection coordinate system. The 3D points visualization in this figure is the same as the corresponding depth image.

3.2.2 Projection with Axis-Aligned Bounding Box

The axis-aligned bounding box (AABB) is another commonly used bounding volume. To provide more robust hand pose estimation, we also project the 3D points onto three orthogonal side planes of AABB, as shown in Figure 3.3 (right). The AABB is the minimum bounding box of which the edges are parallel with the axes of a fixed coordinate system [105]. Here, the fixed coordinate system is the camera coordinate system. The origin of the AABB projection coordinate system is set at the center of the bounding box, and its x , y , z axes are respectively aligned with the x , y , z axes of the camera coordinate system. We have found experimentally that the projected images generated using AABB are complementary to

those generated using OBB, and their integration can further boost the estimation performance.

For generating the projected image on one view, the distances from segmented hand 3D points to the projection plane are first normalized between 0 and 1 (with nearest points set to 0, farthest points in the segmented hand point cloud set to 1). The values of pixels which are not belong to the hand region are set as 1. Then, 3D points are orthographically projected onto the quantized projection plane, and corresponding normalized distances are stored as pixel values of the projected image. If multiple 3D points are projected onto the same pixel region, the smallest normalized distance will be stored as the pixel value. As the projected images may be noisy and may have missing pixels due to self-occlusion in depth images [106], we perform median filtering and morphological operations to further smooth the projected images.

Figure 3.4 presents some examples of projected images on the six views. As can be seen, projected images on different views can reflect different features of hand shapes. The projected images on OBB’s three views are rotation invariant and can reflect front, top and lateral shapes of the hand. The projected images on the AABB’s x - y view are almost the same as input depth images, because the x - y plane in AABB projection coordinate system is aligned with the projection plane of the depth image. In addition, even in the situation where the hand rotation is large, the front hand shape can still be recovered in the projected images, *e.g.* the sixth row’s OBB x - y view and AABB y - z view in Figure 3.4. Although these six views are not mutually orthogonal and there may be some redundancy between OBB’s three views and AABB’s three views, we combine all these six views to make more robust hand pose estimation, and experiments in Section 3.5 will show that the combination method can make more accurate estimation.

3.3 Multi-View Convolutional Neural Networks

3.3.1 Network Architectures

The input projected images are first resized and padded to 96×96 pixels and filtered by local contrast normalization (LCN) [107] to normalize the contrast in the image. The outputs of the network are K heat-maps with 18×18 pixels, of which the intensity indicates the confidence of a joint locating on the 2D position of a specific view.

Multi-view CNNs aim to learn the relations between the projected images and the heat-maps reflecting hand joint locations on each view. As shown in Figure 3.5, we propose to employ the residual network architecture (ResNet) [108] and the fully convolutional network architecture (FCN) [109] for inferring heat-maps from projected images. Residual network is proposed by He *et al.* [108]. The skip connections in ResNet is able to alleviate the gradient vanishing problem during training. Fully convolutional network is proposed by Shelhamer *et al.* [109]. Instead of using fully-connected layers to regress heat-maps, FCN uses deconvolutional layers to generate heat-maps. The number of network parameters is much smaller than the network containing fully-connected layers.

3.3.2 Loss Function for Heat-map Regression

We denote a training sample as (X_t, Φ_t) , where X_t is the depth image in the training dataset, Φ_t is the corresponding joint locations in the camera coordinate system, $t = 1, \dots, T$, T is the number of training samples. The depth image X_t is converted to the multi-view representation $\{I_{t,n}\}_{n=1}^N$ as inputs to multi-view CNNs. The ground truth heat-map $G_{k,n}(\Phi_t)$ for the k -th joint on the n -th view is generated by applying 2D Gaussian centered at the 3D ground truth joint location's 2D projection point on the projection plane. The standard deviation of Gaussian is set to $\sigma = 1.8$ with an output heat-map size of 18×18 , and the Gaussian is

Figure 3.5: (a) Residual Network. (b) Fully Convolutional Network (FCN). We experiment with these two network architectures separately in our experiments. Both networks take a 96×96 projected image as input and generate K heat-maps with the size of 18×18 pixels. The dotted arrows in (a) denote residual connections. ‘BN’ is short for batch normalization, ‘S’ means stride, ‘P’ means padding. For convolutional and deconvolutional layers, the default stride is 1, and the default padding is 0. For max pooling layers, the default stride is the same as kernel size, and the default padding is 0. For simplicity, we do not present stride and padding when using the default values.

normalized to have a sum of 1. In the training stage, we minimize the following objective function:

$$\mathbf{w}_n^* = \arg \min_{\mathbf{w}_n} \sum_{t=1}^T \sum_{k=1}^K \|G_{k,n}(\Phi_t) - \mathcal{H}_k(I_{t,n}, \mathbf{w}_n)\|_F^2, \quad (3.1)$$

where $\|\cdot\|_F$ denotes the Frobenius norm, $\mathbf{w}_n$ denotes network weights for the n -th view, $\mathcal{H}_k$ represents the k -th hand joint’s heat-map output from the CNN regressor.

3.3.3 3D Data Augmentation on Projected Images

Since AABB is sensitive to rotation, we propose to augment the training data by varying global hand orientations. Different with 2D image data augmentation, our proposed 3D data augmentation directly rotates the point cloud of hand in 3D space and then projects the 3D points onto AABB’s three side planes.

The 3D point cloud of hand is rotated around x , y , z axes of the camera coordinate system with rotation angles θ_x , θ_y and θ_z , respectively. The 3D point $\mathbf{p}$ is transformed into $\mathbf{p}'$ after 3D rotation:

$$\mathbf{p}' = \mathcal{R}_x(\theta_x) \cdot \mathcal{R}_y(\theta_y) \cdot \mathcal{R}_z(\theta_z) \cdot \mathbf{p}, \quad (3.2)$$

where $\mathcal{R}_x$, $\mathcal{R}_y$ and $\mathcal{R}_z$ are 3×3 rotation matrices around x , y , z axes, respectively. Figure 3.6 shows an example of 3D data augmentation. As can be seen, the projected images are not simply rotated in 2D image space. 3D rotation is performed on both hand point cloud and corresponding ground truth joint locations. Projected images on AABB's three views are generated from the transformed point cloud. Note that since OBB is rotation invariant, there is no need to augment training data on OBB's three views.

In this work, a transformed training set for AABB's three views is generated by randomly rotating original training samples. The rotation angles θ_x and θ_y are chosen uniformly at random from the interval $[-30^\circ, 30^\circ]$. The rotation angle θ_z is chosen uniformly at random from the interval $[-90^\circ, 90^\circ]$. During the training stage, both the original training set and the transformed training set are used for training CNNs of AABB's three views, in order to make them robust to variations in global hand orientations.

Figure 3.6: An example of 3D data augmentation. The original training sample is shown in the first line, including 3D point cloud, projected images on AABB's three views and ground truth joint locations. The transformed training sample is shown in the second line. In this example, the 3D points are rotated by rotation angles $\theta_x = 10^\circ$, $\theta_y = -20^\circ$ and $\theta_z = 60^\circ$.

3.4 Multi-view Fusion Algorithm

In this section, we describe our proposed multi-view fusion method. The objective for multi-view fusion is to estimate the 3D hand joint locations $\Phi = \{\phi_k\}_{k=1}^K$ from all the N views' heat-maps. We first formulate the multi-view fusion problem as the maximum a posteriori (MAP) estimation, and propose view selection method for suppressing unreliable view's estimation. We then derive the analytical solution to the optimization problem and obtain final estimation of 3D hand joint locations.

3.4.1 Problem Formulation

We estimate the hand joint locations Φ by applying the MAP estimator on the basis of projections $I_1, I_2, \dots, I_N$, which can be viewed as the observations of the 3D hand pose. Given the query hand depth image I_D , we assume that the N projections $I_1, I_2, \dots, I_N$ are independent, conditioned on the joint locations Φ [110]. Although this is not a strict assumption when projection planes are not mutually orthogonal, we will show experimentally that our proposed fusion method based on this assumption is quite effective compared with state-of-the-art methods in Section 3.5. To resolve ambiguous joint estimations, we constrain Φ to a low dimensional subspace. Based on this constraint, we assume that the K hand joint locations are independent conditioned on each view's projection in the low dimensional subspace, and the priori probability $P(\Phi)$ has a constant value when Φ is constrained in the low dimensional subspace. Under these assumptions, the posterior probability of joint locations can be formulated as the product of

the individual estimations from all the N views. The problem to find the optimal hand joint locations Φ^* is thus formulated as follows:

$$\begin{aligned}
\Phi^* &= \arg \max_{\Phi} P(\Phi | I_1, I_2, \dots, I_N) \\
&= \arg \max_{\Phi} P(I_1, I_2, \dots, I_N | \Phi) \cdot P(\Phi) \\
&= \arg \max_{\Phi} \prod_{n=1}^N P(I_n | \Phi) \\
&= \arg \max_{\Phi} \prod_{n=1}^N P(\Phi | I_n) \cdot P(I_n) \\
&= \arg \max_{\Phi} \prod_{k=1}^K \prod_{n=1}^N P(\phi_k | I_n) \\
s.t. \quad &\Phi \in \Omega,
\end{aligned} \tag{3.3}$$

where Φ is constrained to a low dimensional subspace $\Omega \subseteq \Lambda$ in order to resolve ambiguous joint estimations.

The posterior probability of the k -th hand joint location on the n -th view $P(\phi_k | I_n)$ ($k = 1, 2, \dots, K; n = 1, 2, \dots, N$) can be estimated from heat-maps generated by multi-view CNNs. Since the intensity on a heat-map indicates the confidence of a joint locating in the 2D position on the projection plane, we can get the corresponding probability distribution $P(\phi_{k,n} | I_n)$ from the n -th view's k -th heat-map, where $\phi_{k,n}$ is the k -th 3D hand joint location ϕ_k 's 2D projection point on the n -th view's projection plane.

We denote the signed distance from 3D joint location ϕ_k to the n -th view's projection plane as $d_{k,n}$. Thus, ϕ_k can be decomposed into $\phi_{k,n}$ and $d_{k,n}$ which are independent of each other. Assuming that, conditioned on the n -th view's projection I_n , the distribution of the signed distance variable $d_{k,n}$ is uniform, we have:

$$\begin{aligned}
P(\phi_k | I_n) &= P(\phi_{k,n}, d_{k,n} | I_n) \\
&= P(\phi_{k,n} | I_n) P(d_{k,n} | I_n) \\
&\propto P(\phi_{k,n} | I_n).
\end{aligned} \tag{3.4}$$

Figure 3.7: Illusion of multi-view fusion. For illustration purpose, we only present a case having three views for fusion in this figure. The probability distribution $P(\phi_{k,n}|I_n)$ on the n -th view is obtained from its corresponding heat-map ($n = 1, 2, \dots, N$; in this figure, $N = 3$). For a 3D sampling point $\mathbf{p}$, it is projected onto all the N views to get its 2D projection points $\mathbf{p}_n$ and their corresponding heat-map intensities $P(\phi_{k,n} = \mathbf{p}_n | I_n)$. Then, $Q(\phi_k = \mathbf{p}) = \prod_{n=1}^N P(\phi_{k,n} = \mathbf{p}_n | I_n)$.

Thus, the optimization problem in Eq. 3.3 can be transformed into:

$$\begin{aligned}
 \Phi^* &= \arg \max_{\Phi} \prod_{k=1}^K \prod_{n=1}^N P(\phi_{k,n} | I_n) \\
 &= \arg \max_{\Phi} \prod_{k=1}^K Q(\phi_k), \\
 \text{s.t. } \Phi &\in \Omega,
 \end{aligned} \tag{3.5}$$

where $Q(\phi_k) = \prod_{n=1}^N P(\phi_{k,n} | I_n)$ for the k -th hand joint.

Eq. 3.5 indicates that we can get the optimal hand joint locations by maximizing the product of $Q(\phi_k)$ for all the joints. The distribution of $Q(\phi_k)$ can be estimated from sampled values of $Q(\phi_k = \mathbf{p})$, where $\mathbf{p}$ denotes the 3D sampling point. In this work, when using OBB/AABB's three views, the 3D points are uniformly sampled in the corresponding bounding box; when using the combination of OBB and AABB's six views, the 3D points are uniformly sampled in the intersection of OBB and AABB. As shown in Figure 3.7, the 3D sampling point $\mathbf{p}$ is projected onto all the N views to get its 2D projection points $\mathbf{p}_n$ and their corresponding heat-map intensities $P(\phi_{k,n} = \mathbf{p}_n | I_n)$ ($n = 1, 2, \dots, N$). Then, the value of $Q(\phi_k = \mathbf{p})$ for a 3D point $\mathbf{p}$ can be computed by multiplying these intensities.

3.4.2 View Selection

In order to handle the disagreement among multiple views' estimations, if the confidence values of some views are evidently smaller than those of all the other views, these values should be suppressed. For each 3D sampling point $\mathbf{p}$, we define the set of views whose intensities are below a threshold as $\mathbf{U}_{\mathbf{p}} = \{n' | P(\phi_{k,n'} = \mathbf{p}_{n'} | I_{n'}) < \tau_{k,n'}\}$, where $\tau_{k,n'}$ is the threshold determined by the Otsu's thresholding method on the corresponding heat-map. If there is only a small portion of views whose intensities are below the threshold, namely $|\mathbf{U}_{\mathbf{p}}| \leq \varepsilon N$, it means that confidence values provided by these views may be unreliable and the value of $Q(\phi_k = \mathbf{p})$ will be computed from all the other views' intensities:

$$Q(\phi_k = \mathbf{p}) = \left(\prod_{n=1, n \notin \mathbf{U}_{\mathbf{p}}}^N P(\phi_{k,n} = \mathbf{p}_n | I_n) \right)^{\frac{N}{N-|\mathbf{U}_{\mathbf{p}}|}} \quad (3.6)$$

when $|\mathbf{U}_{\mathbf{p}}| \leq \varepsilon N$,

where the power $N/(N - |\mathbf{U}_{\mathbf{p}}|)$ is used for normalization.

If more views' intensities are below the threshold, namely $|\mathbf{U}_{\mathbf{p}}| > \varepsilon N$, we no longer suppress the small intensities and the value of $Q(\phi_k = \mathbf{p})$ will be assigned to zero. In our implementation, the coefficient ε is set to $\varepsilon = 1/5$. Thus, when we use OBB/AABB's three views, namely $N = 3$, no views' intensities will be suppressed; when we use the combination of OBB and AABB's six views, namely $N = 6$, at most one view's intensity will be suppressed. By applying the method of view selection, information in heat-maps can be more effectively utilized, thus the estimation will be more robust and accurate.

3.4.3 Solution to the Optimization Problem

For simplicity of the optimization problem, the distribution of $Q(\phi_k)$ is approximated as a 3D Gaussian distribution $\mathcal{N}(\mu_k, \Sigma_k)$, where μ_k is the weighted mean vector, Σ_k is the weighted covariance matrix. These parameters of the Gaussian distribution can be estimated from the sampled values of $Q(\phi_k = \mathbf{p})$:

$$\mu_k = \sum_{\mathbf{p}} w_{\mathbf{p}}^{(k)} \mathbf{p}, \quad \Sigma_k = \sum_{\mathbf{p}} w_{\mathbf{p}}^{(k)} (\mathbf{p} - \mu_k) (\mathbf{p} - \mu_k)^T, \quad (3.7)$$

where $w_{\mathbf{p}}^{(k)} = Q(\phi_k = \mathbf{p}) / \sum_{\mathbf{p}'} Q(\phi_k = \mathbf{p}')$ is the weight of sampling point $\mathbf{p}$ for joint k .

Based on above assumptions and derivations, the optimization problem in Eq. 3.5 can be approximated as follows:

$$\begin{aligned} \Phi^* &= \arg \max_{\Phi} \sum_{k=1}^K \log Q(\phi_k) \\ &= \arg \max_{\Phi} \sum_{k=1}^K \log \mathcal{N}(\mu_k, \Sigma_k) \\ &= \arg \min_{\Phi} \sum_{k=1}^K (\phi_k - \mu_k)^T \Sigma_k^{-1} (\phi_k - \mu_k) \\ s.t. \quad \Phi &= \sum_{m=1}^M \alpha_m \mathbf{e}_m + \mathbf{u}, \end{aligned} \quad (3.8)$$

where Φ is constrained to take the linear form. In order to learn the low dimensional subspace Ω of hand configuration constrains, PCA is performed on joint locations in the training dataset during the training stage [7]. $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_M]$ are the principal components, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_M]^T$ are the coefficients of the principal components, $\mathbf{u}$ is the empirical mean vector, and $M \ll 3 \times K$.

3.4.3.1 Derivation of the Optimal Solution

The optimization problem is formulated as follows:

$$\begin{aligned} \Phi^* &= \arg \min_{\Phi} \sum_k (\phi_k - \mu_k)^T \Sigma_k^{-1} (\phi_k - \mu_k) \\ s.t. \quad \Phi &= \sum_{m=1}^M \alpha_m \mathbf{e}_m + \mathbf{u} = \mathbf{E}\alpha + \mathbf{u} \end{aligned} \quad (3.9)$$

Let $R(\Phi) = \sum_k (\phi_k - \mu_k)^T \Sigma_k^{-1} (\phi_k - \mu_k)$, which is in the quadratic form of the variable α , the optimal solution of α can be obtained by setting the derivative of R with respect to α to zero.

$$\begin{aligned} \frac{\partial R(\Phi)}{\partial \alpha} &= \frac{\partial R(\Phi)}{\partial \Phi} \cdot \frac{\partial \Phi(\alpha)}{\partial \alpha} \\ \therefore \frac{\partial R(\Phi)}{\partial \phi_k} &= \frac{\partial}{\partial \phi_k} \left[(\phi_k - \mu_k)^T \Sigma_k^{-1} (\phi_k - \mu_k) \right] \\ &= 2(\phi_k - \mu_k)^T \Sigma_k^{-1} \end{aligned} \quad (3.10)$$

$$\therefore \frac{\partial R(\Phi)}{\partial \Phi} = 2 \begin{bmatrix} \Sigma_1^{-1} (\phi_1 - \mu_1) \\ \vdots \\ \Sigma_k^{-1} (\phi_k - \mu_k) \\ \vdots \\ \Sigma_K^{-1} (\phi_K - \mu_K) \end{bmatrix}^T \quad (3.11)$$

$$\therefore \frac{\partial \Phi(\alpha)}{\partial \alpha} = \frac{\partial}{\partial \alpha} (\mathbf{E}\alpha + \mathbf{u}) = \mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_M] \quad (3.12)$$

$$\therefore \frac{\partial R(\Phi)}{\partial \alpha} = 2 \begin{bmatrix} \Sigma_1^{-1} (\phi_1 - \mu_1) \\ \vdots \\ \Sigma_k^{-1} (\phi_k - \mu_k) \\ \vdots \\ \Sigma_K^{-1} (\phi_K - \mu_K) \end{bmatrix}^T \cdot [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_M] = \mathbf{0} \quad (3.13)$$

Thus, we can get M linear equations for M unknown variables $\alpha_1, \alpha_2, \dots, \alpha_M$:

$$\sum_{k=1}^K \left[\left(\sum_{j=1}^M \alpha_j \mathbf{e}_{j,k}^T + \mathbf{u}_k^T - \boldsymbol{\mu}_k^T \right) \boldsymbol{\Sigma}_k^{-1} \mathbf{e}_{i,k} \right] = 0, \quad (3.14)$$

$$i = 1, 2, \dots, M$$

$$\therefore \sum_{j=1}^M \left[\left(\sum_{k=1}^K \mathbf{e}_{j,k}^T \boldsymbol{\Sigma}_k^{-1} \mathbf{e}_{i,k} \right) \alpha_j \right] = \sum_{k=1}^K (\boldsymbol{\mu}_k - \mathbf{u}_k)^T \boldsymbol{\Sigma}_k^{-1} \mathbf{e}_{i,k} \quad (3.15)$$

Let $\mathbf{A}\boldsymbol{\alpha} = \mathbf{b}$, then:

$$\mathbf{A}_{ij} = \sum_{k=1}^K \mathbf{e}_{j,k}^T \boldsymbol{\Sigma}_k^{-1} \mathbf{e}_{i,k}, \quad \mathbf{b}_i = \sum_{k=1}^K (\boldsymbol{\mu}_k - \mathbf{u}_k)^T \boldsymbol{\Sigma}_k^{-1} \mathbf{e}_{i,k} \quad (3.16)$$

$$i, j = 1, 2, \dots, M$$

At last, we can get the solution for the optimization problem: $\boldsymbol{\alpha}^* = \mathbf{A}^{-1}\mathbf{b}$.

As proved in this section, given the linear constraints of Φ , the optimal coefficient vector $\boldsymbol{\alpha}^* = [\alpha_1^*, \alpha_2^*, \dots, \alpha_M^*]^T$ is:

$$\boldsymbol{\alpha}^* = \mathbf{A}^{-1}\mathbf{b}, \quad (3.17)$$

where $\mathbf{A}$ is an $M \times M$ symmetric matrix, $\mathbf{b}$ is an M -dimensional column vector:

$$\mathbf{A}_{ij} = \sum_k \mathbf{e}_{j,k}^T \boldsymbol{\Sigma}_k^{-1} \mathbf{e}_{i,k}, \quad \mathbf{b}_i = \sum_k (\boldsymbol{\mu}_k - \mathbf{u}_k)^T \boldsymbol{\Sigma}_k^{-1} \mathbf{e}_{i,k},$$

$$\mathbf{e}_i = [\mathbf{e}_{i,1}^T, \mathbf{e}_{i,2}^T, \dots, \mathbf{e}_{i,K}^T]^T; \quad \mathbf{u} = [\mathbf{u}_1^T, \mathbf{u}_2^T, \dots, \mathbf{u}_K^T]^T; \quad i, j = 1, 2, \dots, M.$$

The optimal joint locations Φ^* are reconstructed by back-projecting the optimal coefficients $\boldsymbol{\alpha}^*$ in the subspace Ω to the original joint space Λ :

$$\Phi^* = \sum_{m=1}^M \alpha_m^* \mathbf{e}_m + \mathbf{u}. \quad (3.18)$$

To sum up, the proposed method for multi-view fusion consists of two main steps. The first step is to estimate the parameters of Gaussian distribution $Q(\phi_k)$

for each joint using all the N views' heat-maps. Unreliable intensities of some views will be suppressed when calculating the sampled value of $Q(\phi_k = \mathbf{p})$. The second step is to calculate the optimal coefficients α^* and reconstruct the optimal joint locations Φ^* . Since an analytical solution has been derived, the 3D joint locations can be inferred efficiently using the proposed multi-view fusion algorithm.

3.5 Experiments

We conduct self-comparisons and comparisons with state-of-the-art methods on two hand pose datasets: MSRA2015 dataset [1] and NYU dataset [2]. We also conduct corss-dataset experiments on MSRA2014 hand pose dataset [66].

We employ three metrics to evaluate the estimation performance. The first metric is the mean error distance for each hand joint across all test frames, which is a standard evaluation metric. The second metric is the proportion of “good” test frames among all test frames. A test frame is regarded as “good” only when all estimated joint locations are within a maximum allowed distance from the ground truth, namely the error threshold. This worst case accuracy proposed in [111] is a very strict criterion. The third metric is the proportion of joints within an error threshold among all test joints [8].

3.5.1 Implementation Details

Our proposed multi-view CNNs are implemented within the Torch7 [112] framework. When training the multi-resolution network in [15], we use the stochastic gradient descent algorithm with learning rate of 0.2, batch size of 64, momentum of 0.9 and weight decay of 0.0005. When training ResNet and FCN, we use the RMSprop algorithm with learning rate of 0.001, batch size of 64, epsilon of 0.01. Training is stopped after 50 epochs to prevent overfitting. All experiments are conducted on a computer with two Intel Core i7-5930K processors, 64GB of RAM and two Nvidia Tesla K80 GPUs each having 12GB of GPU memory.

3.5.2 Self-Comparison

3.5.2.1 Single View CNN Versus Multi-View CNNs

To evaluate the superiority of the multi-view method over the single view method, we implement the single view CNN-based approach. In our implementation, only projected images on the x - y plane of OBB projection coordinate system are fed into the CNN, which is similar to the method in [2]. We can only estimate x and y coordinates of hand joints from output heat-maps using the 2D Gaussian fitting method as that in [2]. The z coordinate can be inferred using the pixel value of the projected image. If the estimated 2D position of a hand joint is on the background of the projected image, the z coordinate will be set as zero in the OBB projection coordinate system rather than the maximum depth value, which can lower the estimation error on z direction. As presented in Figure 3.8 (left and middle), the multi-view regression approach significantly outperforms the single view regression approach.

Figure 3.8: Self-comparison of different methods on MSRA2015 dataset [1] using OBB’s three views. **Left:** the mean error distance for each joint across all the test frames (R: root, T: tip). **Middle:** the proportion of good test frames in the entire test frames over different error thresholds. **Right:** the influence of different numbers of principal components used in hand pose constraints on the estimation accuracy.

3.5.2.2 Evaluating Fusion Methods

To evaluate the effectiveness of our proposed optimal fusion method, we implement the average fusion method as baseline, which can be regarded as a simplified alternative of the proposed optimal fusion method. This method first estimates 2D projection points of the 3D hand joint location on x - y , y - z and z - x planes of the OBB projection coordinate system by fitting Gaussian models on corresponding heat-maps. The 3D hand joint location is determined by averaging x , y and z coordinates, which are obtained from three views' 2D projection points, in the OBB projection coordinate system. As can be seen in Figure 3.8 (left and middle), the mean error over all joints of the optimal fusion method is 13.1mm on MSRA2015 dataset, while that of the average fusion method is 15.8mm; for the worst case accuracy, the optimal fusion method performs better than the average fusion method when the error threshold is smaller than 50mm, but is a little bit worse than the average fusion method when the error threshold is larger than 50mm. The reason may be that the average fusion method takes the average among ambiguous estimations, which may not deviate from the right estimation very large. The optimal fusion method is able to choose the optimal one among ambiguous estimations. But once this method chooses the wrong estimation, the estimation error will be very large. However, high accuracy at small error threshold should be more favorable. The optimal fusion method is overall better than the average fusion method and we adopt the optimal fusion method in the following experiments.

An experimental example of the ambiguous estimation is presented in Figure 3.9, where the tip of index finger is very likely to be confused with the tip of little finger. As shown in this figure, the single view regression method only exploits the heat-map of OBB's x - y view containing two hotspots, thus estimates the location of index fingertip with large error distance. However, the multi-view optimal fusion method fuses heat-maps of three views and estimates the 3D location with high accuracy. The estimation of the multi-view average fusion method is between those of the above two methods, as this method underutilizes the information of heat-maps.

Additionally, we evaluate the impact of different numbers of principal components M used for the constraint in Eq. 3.8 on the worst case accuracy under

Figure 3.9: A case of ambiguous estimation. **Top-left:** 3D point cloud with ground truth and estimated 3D locations of the tip of index finger. **Top-right:** projected images on OBB’s three views. **Bottom-right:** heat-maps of OBB’s three views. The ground truth and estimated 3D locations of the tip of index finger are projected onto three views and their heat-maps for comparison. Lines indicate offsets between estimated joint locations and the ground truth.

different error thresholds. As can be seen in Figure 3.8 (right), it is suitable to use 35 principal components for constraint considering the estimation accuracy. We set $M = 35$ for all the other experiments.

3.5.2.3 Impact of Heat-map Resolution

We evaluate the impact of the heat-map resolution on accuracy and real-time performance. In this experiment, we apply the FCN architectures with view selection and data augmentation for generating heat-maps with different resolutions, *i.e.*, 9×9 , 18×18 and 36×36 . To make fair comparison, we design the FCNs with comparable numbers of network parameters for different heat-map resolutions. As shown in Figure 3.10, when the error threshold is smaller than 30mm, the method with 36×36 heat-maps performs best, and the method with 9×9 heat-maps performs worst. However, when the error threshold is larger than 30mm, the method with 36×36 heat-maps performs worst. Moreover, the method with 36×36 heat-maps is time-consuming, which runs at 18.7fps and cannot achieve real-time performance. Balancing between the runtime and the estimation accuracy, we choose 18×18 as the heat-map resolution.

Figure 3.10: Self-comparison of different heat-map resolutions on NYU [2] hand pose dataset. The mean error distances over all joints and the average frame rates are shown in the legends.

3.5.2.4 Effectiveness of Combination OBB with AABB

We experiment with the hand pose estimation method using OBB/AABB’s three views as well as the method using both OBB and AABB’s six views. As shown in Figure 3.11, compared with the method using OBB’s three views, the method using AABB’s three views performs almost the same on MSRA2015 dataset, and performs worse on NYU dataset. But the method using both OBB and AABB’s six views outperforms the first two methods over most error thresholds even without using the view selection method. These results demonstrate that the projected images of OBB and AABB’s six views are complementary to each other, and the fusion of six views’ heat-maps can produce more accurate and robust results.

3.5.2.5 Effectiveness of View Selection

When fusing heat-maps from multiple views, our method selects views with reliable estimations and suppresses unreliable estimations. As shown in Figure 3.11, when using the combination of OBB and AABB’s six views, the view selection method will further improve the estimation accuracy, which indicates that it is effective to deal with the disagreement among multiple views’ estimations by using our proposed view selection method.

Figure 3.11: Evaluation of combining OBB and AABB’s six views, view selection method, 3D data augmentation and different network architectures on MSRA2015 dataset [1] and NYU dataset [2]. The mean error distances over all joints of different methods are shown in the legend titles.

3.5.2.6 Effectiveness of 3D Data Augmentation

In Section 3.3.3, we augment the training data for AABB’s three views by randomly rotating the 3D point cloud. As shown in Figure 3.11, when applying 3D data augmentation, the estimation accuracy is slightly better than the method without using data augmentation. If the variation in hand global orientations is larger for the test frames, the improvement achieved by 3D data augmentation will be more evident.

3.5.2.7 Comparison of Network Architectures

In all of the above experiments, we employ the multi-resolution network in [15] to estimate the heat-maps. Here, we experiment with the ResNet and FCN. As shown in Figure 3.11, both the ResNet and FCN can improve the estimation accuracy, especially on the NYU dataset, which shows that both networks have good generalization ability. Moreover, the FCN has fewer parameters (24 million on both datasets) than the multi-resolution network (99 million on the MSRA dataset and 56 million on the NYU dataset) and ResNet (93 million on the MSRA dataset and 58 million on the NYU dataset). Thus, the model size of FCN is smaller. But the estimation accuracy of ResNet is slightly better than that of FCN. In following comparisons with state-of-the-art methods, we present results of the ResNet due to its better estimation accuracy.

Figure 3.12: Comparison with state-of-the-art approaches on MSRA2015 dataset [1]. **Left:** the proportion of good test frames in the entire test frames over different error thresholds. **Middle & right:** the mean error distance over different yaw and pitch angles of the viewpoint. Some curves are cropped from corresponding figures reported in [1, 76, 83].

3.5.3 Comparison with State-of-the-Art

In this section, we compare our methods with state-of-the-art methods. We denote the method using OBB’s three views and multi-resolution network as *MVCNN-OBB*, and the method using OBB and AABB’s six views with the ResNet architecture, view selection and data augmentation as *MVCNN-Hybrid*.

3.5.3.1 Comparison on MSRA2015 Dataset

On MSRA2015 dataset, we compare our multi-view CNN-based method with three state-of-the-art methods: the random forest based hierarchical regression method [1], the JMFC based collaborative filtering method [76] and the local surface normals based method with finger jointly regression and pose classification [83]. Note that the first two methods have been validated to be superior to methods in [3, 4, 27]. Thus, we indirectly compare with these methods.

As shown in Figure 3.12 (left), our MVCNN-Hybrid method achieves the best performance when the error threshold is larger than 5mm, and our MVCNN-OBB method achieves the second best performance when the error threshold is between 15mm and 30mm. When the error threshold is 5mm, the good frame proportions of our methods are slightly inferior to those of the hierarchical regression method [1] and the JMFC based method [76]. This may be induced by the relatively low resolution of heat-maps (18×18) adopted in our methods.

We evaluate the mean error distances over different yaw and pitch angles of these methods. As presented in Figure 3.12 (middle and right), the mean error distances of our MVCNN-OBB method and MVCNN-Hybrid method are about 2mm and 5mm smaller than those of the hierarchical regression method [1] over all the yaw and pitch angles, respectively. Moreover, our methods are more robust to the pitch angle variation with smaller stand deviations (0.64mm for MVCNN-OBB and 0.58mm for MVCNN-Hybrid) than the hierarchical regression method (0.79mm) [1].

3.5.3.2 Comparison on NYU Dataset

On NYU dataset, we first compare the worst case accuracies of our proposed multi-view CNN-based methods with five state-of-the-art methods. The first method is the single view CNN-based heat-map regression method proposed in [2] which directly adopts the depth image as the input of the network (denoted as *Single View*). For comparison, the estimated 2D joint location in the depth image is converted to 3D location with the information of its corresponding depth value. The second method is the CNN-based direct hand pose estimation with a prior proposed in [94] (denoted as *Prior*). The third method is the CNN-based hand pose estimation using a feedback loop proposed in [5] (denoted as *Feedback*). The fourth method is the CNN-based hand model parameters regression proposed in [95] (denoted as *Model-based*). The last method is the deep feature based matrix completion

Figure 3.13: Comparison with state-of-the-art approaches on NYU dataset [2]. **Left:** the proportion of good test frames in the entire test frames over different error thresholds. **Right:** the proportion of joints within different error thresholds. Some curves cropped from corresponding figures reported in [5, 88, 94–96].

method proposed in [96] (denoted as *DeepHand*). As shown in Figure 3.13 (left), our MVCNN-Hybrid method significantly outperforms these five state-of-the-art methods when the error threshold is larger than 10mm, and our MVCNN-OBB method achieves the second best performance when the error threshold is between 20mm and 45mm.

In order to make a fair comparison with the spatial attention network based hierarchical hybrid method proposed in [88] (denoted as *Hybrid_Hier_SA*), we evaluate the proportion of joints within different error thresholds on the subset containing 11 hand joints following the experimental setting in [88] (removing palm joints except the root joint of thumb). As shown in Figure 3.13 (right), our MVCNN-Hybrid method outperforms the methods in [5, 88, 94] over all the error thresholds, and our MVCNN-OBB method achieves the second best performance when the error threshold is larger than 10mm.

3.5.4 Cross-Dataset Experiment

We conduct a cross-dataset experiment to further validate the generalization ability of our proposed multi-view CNN-based hand pose estimation method. In this experiment, we aim at adapting the multi-view CNNs trained on the source MSRA2015 dataset [1] to the target MSRA2014 dataset [66].

We train the multi-view CNNs on all the nine subjects in MSRA2015 dataset [1]. The fully trained multi-view CNNs are evaluated on all the six subjects in MSRA2014 dataset [66]. Following the evaluation criterion in [66], we only evaluate the average estimation errors for the wrist center and five fingertips. As shown in Table 3.1, we compare our MVCNN-OBB method and MVCNN-Hybrid method with model-based tracking approaches presented in [66], which are FORTH [27], PSO [66], ICP [113], ICP-PSO [66] and ICP-PSO* (ICP-PSO with finger based initialization) [66].

As reported by [66], these model-based tracking methods require a carefully calibrated hand model for each subject’s hand, and these methods rely on the temporal information for tracking. Particularly, these methods utilize ground truth information for the first frame initialization. However, our methods do not utilize these information, thus are more flexible for different subjects and are robust to

Table 3.1: Average Estimation Errors (in mm) of 6 Subjects for 7 Methods Evaluated on MSRA2014 Dataset [66].

Subject	1	2	3	4	5	6	Avg
FORTH	35.4	19.8	27.3	26.3	16.6	46.2	28.6
PSO	29.3	14.8	40.2	17.3	16.2	24.3	23.6
ICP	29.9	20.7	30.8	23.9	18.5	32.8	26.1
ICP-PSO	10.1	24.1	13.0	12.8	11.9	20.0	15.3
ICP-PSO*	8.6	7.4	9.8	10.4	7.8	11.7	9.2
MVCNN-OBB	30.1	19.7	24.3	19.9	21.8	20.7	22.8
MVCNN-Hybrid	21.8	16.3	16.7	16.3	14.7	16.1	17.0

estimation failure. Despite these unfavorable conditions, both of our MVCNN-OBB method and MVCNN-Hybrid method still outperform FORTH, PSO and ICP methods, as presented in Table 3.1, which demonstrates that our methods have good generalization ability. It is not surprising that our methods are inferior to ICP-PSO and ICP-PSO*, since ICP-PSO and ICP-PSO* are tracking methods that require a carefully calibrated hand model and ground truth for first frame initialization, while our methods do not require any hand model and do not utilize any temporal information for tracking or ground truth for initialization. Furthermore, we conduct this experiment on cross-dataset that is more challenging. It is worth noting that the average estimation error of our MVCNN-Hybrid method is only 1.7mm worse than that of the ICP-PSO method.

3.5.5 Qualitative Results

Figure 3.14 shows qualitative results of the MVCNN-OBB method and the MVCNN-Hybrid method on several challenging test frames in MSRA2015 dataset [1] and NYU dataset [2]. As can be seen, the estimation accuracy of the MVCNN-Hybrid method is generally better than that of the MVCNN-OBB method. It is worth noting that even though some depth images in NYU dataset are very noisy and some frames are incomplete, *e.g.*, the 6th and 10th columns in Figure 3.14, our proposed multi-view CNN-based methods are still able to make accurate estimations of 3D hand joint locations from these depth images.

Figure 3.14: Qualitative results on MSRA2015 dataset [1] and NYU dataset [2] of two methods: the MVCNN-OBB method (in the first line) and the MVCNN-Hybrid method (in the second line). The ground truth hand joint locations are shown in the last line. We show hand joint locations and bones with the point cloud. Different hand joints and bones are visualized using different colors. This figure is best viewed in color.

Figure 3.15: Qualitative results for testing in real scenarios on different subjects with different hand sizes and hand poses. For each subject, the first line are the depth images captured by the SoftKinetic's DepthSense camera; the second line is the segmented hand depth images and the estimated hand joint locations. Different hand joints and bones are visualized using different colors. This figure is best viewed in color.

We additionally evaluate our pre-trained multi-view CNN models in real scenarios. We train the multi-view CNN models on all the 9 subjects of the MSRA2015 dataset in [1] and apply these models to perform real-time hand pose estimation. As shown in Figure 3.15 (the 1st line), the depth images are captured by the SoftKinetic’s DepthSense camera which is different with the Intel’s Creative Interactive Gesture Camera used in MSRA2015 dataset [1]. We experiment on three subjects, who are not included in MSRA2015 dataset [1], with different hand sizes and hand poses. The qualitative results of hand pose estimation using our MVCNN-Hybrid method are shown in Figure 3.15 (the 2nd line). As can be seen, our proposed multi-view CNN-based hand pose estimation method is tolerant for different hand poses and hand sizes, and it is even tolerant for different types of depth cameras of which the noise distribution may be different. This experiment further demonstrates the good generalization ability of our proposed multi-view CNN-based method.

3.5.6 Runtime

The runtime of our MVCNN-OBB method is 12.2ms in average, including 1.8ms for depth image preprocessing and multi-view projection, 6.8ms for multi-resolution networks forward propagation and 3.6ms for multi-view fusion. Therefore, the MVCNN-OBB method runs in real-time at about 82fps. The runtime of our MVCNN-Hybrid method is 16.8ms in average, including 3.2ms for depth image preprocessing and multi-view projection, 9.3ms for ResNets forward propagation and 4.3ms for multi-view fusion. Therefore, the MVCNN-Hybrid method runs in real-time at about 60fps. Note that tasks of multi-view projection and multi-view fusion are executed on CPU with parallelism, and the task of CNNs forward propagation is executed on GPU with parallelism. Although the MVCNN-Hybrid method is slower than the MVCNN-OBB method, the MVCNN-Hybrid method can achieve more accurate and robust results than the MVCNN-OBB method and it can still run fast in real-time.

3.6 Summary

In this chapter, we have presented a multi-view CNNs-based approach for robust 3D hand pose estimation. The 3D point cloud generated from the depth image is projected onto both OBB and AABB's six views. The multi-view CNNs are trained to map projected images to heat-maps representing the probability distributions of hand joints on projected images. Two network architectures are proposed using ResNet and FCN which have good generalization ability. 3D data augmentation is performed on training CNNs of AABB's three views to make them more robust to various hand orientations. Heat-maps from different views are fused to make the optimal estimation of 3D hand joint locations. Furthermore, we have proposed a view selection method to suppress unreliable information in multi-view heat-maps. Experimental results are presented to show the superior performance and good generalization ability of our proposed methods.

Chapter 4

3D CNN-based 3D Hand Pose Estimation

4.1 Overview

Many recent works on hand pose estimation have achieved good performance due to the success of Convolutional Neural Networks (CNNs) [2, 5, 15, 88–90, 94–96, 114] and the availability of large hand pose datasets [1–3]. Most of these methods directly take the depth image as input to 2D CNNs which output heat-maps [2], 3D joint locations [5, 88, 90, 94, 96] (Figure 4.1a) or hand model parameters [95]. Nevertheless, we argue that image-based features extracted by 2D CNNs are not directly suitable for 3D hand pose estimation due to the lack of 3D spatial information. For example, in [5], the initial result of 2D CNN is poor, and it is iteratively refined by a feedback loop to incorporate 3D information from a generative model.

In Section 3, we project the depth image onto three views and apply multi-view CNNs to regress multiple views’ heat-maps. However, the multi-view CNNs still cannot fully exploit 3D spatial information in the depth image, since the projection from 3D to 2D will lose certain information. Although increasing the number of views may improve the performance, the computational complexity will increase when using more views.

Figure 4.1: Different schemes for 3D hand pose estimation. (a) 2D CNN taking depth image as input and regressing 3D joint locations directly. (b) We use 3D CNN which takes volumetric representations as input and regresses a lower dimensional representation of 3D joint locations.

In this chapter, we propose a 3D CNN-based hand pose estimation approach that can capture the 3D spatial structure of the input and accurately regress full 3D hand pose in a single pass, as illustrated in Figure 4.1b. Specifically, human hand is first segmented from the depth image; the 3D point cloud of the hand is encoded as 3D volumes storing the projective Directional Truncated Signed Distance Function (D-TSDF) [115] values, which are then fed into a 3D convolutional neural network. We design a 3D deep dense network to boost the learning ability of the network. The output of this network is a lower dimensional representation of 3D hand joints' relative locations in the 3D volume. By performing PCA reconstruction and coordinate transformations, we can finally obtain the 3D hand joint locations in the camera's coordinate system. Benefiting from 3D features extracted by 3D convolutions, our method is able to understand the hand pose structure in 3D space and infer 3D hand joint locations efficiently and robustly.

Compared to previous CNN-based methods for hand pose estimation, our proposed 3D CNN-based method has the following advantages:

- Our proposed 3D CNN is capable of learning 3D features from the 3D volumetric representation for accurate 3D hand pose estimation. Compared to 2D CNN-based methods that regress 3D joint locations from 2D features [5, 88, 94, 116], 3D CNN can directly regress 3D joint locations from 3D features in a single pass. This not only achieves superior estimation accuracy, but also avoids the time-consuming iterative refinement process.

- Our proposed 3D CNN can be easily trained in an end-to-end manner. Both the 3D shallow plain network and the 3D deep dense network we designed in this chapter can run at real-time speed on a single GPU.
- Our proposed method is robust to variations in hand sizes and global orientations, since we perform 3D data augmentation on the training set. Different from traditional data augmentation that performs 2D transformations on 2D images, our proposed 3D data augmentation applies 3D transformations on 3D point clouds, thus can better enrich the training data in 3D space.

The contributions of this chapter are summarized as follows:

- We have proposed a 3D CNN-based hand pose estimation method that is able to capture 3D features of the hand from 3D volumes. Experimental results have shown that our proposed method is able to accurately regress 3D hand pose from the depth images.
- We have proposed leveraging the complete hand surface as intermediate supervision for learning 3D hand pose from depth images. Experimental results have shown that, with the intermediate hand surface completion step, the estimation accuracy of 3D hand pose can be further improved.
- We have investigated the performance of deep neural network architectures [108, 117] when applied in our proposed 3D CNN-based framework for 3D hand pose estimation to improve the learning ability. Experimental results have shown that the 3D deep dense network can achieve better performance than the 3D shallow plain network.
- To better understand 3D CNNs, we have visualized the input patterns that produce given activations in the 3D feature volumes. Local and global 3D structures of hand have been observed in these patterns.
- We have conducted extensive self-comparison experiments and have compared with state-of-the-art methods. We have also conducted a cross-dataset experiment and qualitatively compared with the Intel RealSense SDK [118]. Experimental results have shown that our method can achieve good performance in real-time and has good generalization ability.

4.2 Volumetric Representations

Our proposed 3D CNN-based hand pose estimation method takes a volumetric representation as the input of the neural network. The objective for encoding the observed hand depth image as a volumetric representation is to generate 3D volumes providing sufficient and meaningful information of the hand in 3D space from the depth image in real-time. The 3D volumes will be fed into the 3D CNNs to learn 3D features for subsequent 3D hand joint location regression.

The input depth image of hand is first converted to a set of 3D points, which is denoted as $\mathcal{P} \subset \mathbb{R}^3$, as shown in the first two rows of Figure 4.2. To create a 3D volume containing $M \times M \times M$ voxels, we first build an axis-aligned bounding box (AABB) for the 3D hand points. AABB is the minimum bounding box of which x , y , z axes are respectively aligned with x , y , z axes of the camera’s coordinate system. The 3D volume’s center is set at the center of AABB, and its faces are set to be parallel to those of AABB. The edge length of a voxel is set as:

$$l_{voxel} = \max \{l_x, l_y, l_z\} / M, \quad (4.1)$$

where l_x , l_y , l_z are AABB’s edge lengths along three directions; M is the volume resolution value. We denote the value of a voxel $\mathbf{v}$ as $F(\mathbf{v})$ that can be determined by using the occupancy model (Section 4.2.1) or the truncated signed distance function (Section 4.2.2).

4.2.1 Occupancy Grid

The occupancy grid is a binary grid representing occupied and unoccupied voxels in the 3D volume, as shown in the 3rd row of Figure 4.2. Thus, the voxel value is determined as:

$$F(v) = \begin{cases} 1 & \exists \mathbf{p} \in \mathcal{P}, \text{ s.t. } D_{Che}(\mathbf{v}_c, \mathbf{p}) \leq l_{voxel}/2 \\ 0 & \text{otherwise} \end{cases}, \quad (4.2)$$

where $D_{Che}(\mathbf{p}, \mathbf{q}) = \max_i (|\mathbf{p}_i - \mathbf{q}_i|)$ is the Chebyshev distance; $\mathbf{v}_c$ is the center of the voxel $\mathbf{v}$.

If the input is a 3D CAD model where the 3D information is fully known, the occupancy grid is sufficient to represent the complete 3D model. However, in our problem, the input is a 2.5D depth image which only captures the observed surface points from the view of camera, which is an incomplete 3D shape of the hand. The occupancy grid cannot differentiate voxels before and behind the observed surface.

4.2.2 TSDF Volumes

In the accurate Truncated Signed Distance Function (TSDF) based volume, each voxel stores the truncated signed distance from the voxel center to the closest surface point. The value of a voxel $\mathbf{v}$ is computed as:

$$F(\mathbf{v}) = \min \{ \max \{ d(\mathbf{v}_c) / \mu, -1 \}, 1 \}, \quad (4.3)$$

where $d(\mathbf{v}_c)$ is the signed distance from the voxel center $\mathbf{v}_c$ to the closest surface point; when the voxel center's depth value is smaller than the depth value of the closest surface point, its sign is positive; otherwise, its sign is negative. μ is the truncated distance, which is set as $3 \times l_{voxel}$ here.

However, it is time-consuming to compute accurate TSDF, as all surface points must be checked to find the closest point for each voxel in the 3D volume. For real-time considerations, the projective TSDF [119], where the closest point is found only on the line of sight in the camera frame, should be used. It can be computed

Figure 4.2: Visualization of different volumetric representations. For the last five rows, we only visualize voxels of which values are less than 1 and larger than -1 by using the color map shown in this figure. The volume resolution is $32 \times 32 \times 32$.

efficiently in parallel on a GPU. Since the projective TSDF is an approximation of the accurate TSDF, some information is inaccurate or lost in the projective TSDF. In this work, we apply the projective Directional TSDF (D-TSDF) [115] to encode more information in the volumetric representation, in which each voxel stores a 3D offset vector $[dx, dy, dz]$ to the closest point instead of a scalar distance.

Figure 4.2 (the last five rows) shows some examples of accurate TSDF volumes, projective TSDF volumes and projective D-TSDF volumes with different hand

poses. As can be seen, in accurate TSDF volumes, the value of TSDF increases when moving from the observed surface to free space and decreases when moving to the occluded space. In projective TSDF volumes, the value of TSDF varies continuously along the line of sight, which keeps positive in front of the observed surface and negative behind the observed surface. The projective D-TSDF volume of z direction is almost the same as the projective TSDF volume, since the z direction is close to the line of sight. Projective D-TSDF volumes of x, y directions can provide more information to compensate the inaccuracy of the projective TSDF volume. Experiments in Section 4.6.2.2 will show that the projective D-TSDF is computationally efficient and can improve the estimation accuracy.

We need to strike a balance between prediction accuracy and computational cost to determine the volume resolution M . If the volume resolution is too large, it will be time-consuming and memory intensive to compute the volumetric representation. If the volume resolution is too small, the volumetric representation cannot give sufficient information for 3D hand pose estimation. In this work, M is chosen as 32, which proves to have good estimation accuracy and computational efficiency for 3D hand pose estimation according to the experiments in Section 4.6.2.1.

Figure 4.3: Plain and dense network architectures. Both networks take three $32 \times 32 \times 32$ projective D-TSDF volumes as input and output F elements. In (a), all the 3D convolutional layers have stride 1 and no padding. In (b), the growth rate of the 3D deep dense network is 32; L in each 3D dense block denotes the number of dense unit; each dense unit consists of a sequence of layers: BN-ReLU-3D Conv ($1 \times 1 \times 1$)-BN-ReLU-3D Conv ($3 \times 3 \times 3$); each transition layer reduces the number of feature maps by half using $1 \times 1 \times 1$ 3D convolution and downsamples the feature map using average pooling. ‘BN’ denotes the batch normalization layer, ‘FC’ denotes the fully-connected layer, and ‘DP’ denotes the dropout layer.

4.3 3D Convolutional Neural Networks

4.3.1 Network Architectures

Our proposed 3D CNN takes three volumes of the projective D-TSDF as inputs and output a vector containing F elements. As shown in Figure 4.3, we design two network architectures: the 3D shallow plain network and the 3D deep dense network.

4.3.1.1 3D Shallow Plain Network

The 3D shallow plain network contains three 3D convolutional layers and three fully connected layers, as shown in Figure 4.3a. For the three 3D convolutional layers, the kernel sizes are 5^3 , 3^3 and 3^3 , all with stride 1 and no padding. The first two 3D convolutional layers are followed by 3D max pooling layers with kernel size 2^3 , stride 2 and no padding. After 3D feature extraction by 3D convolutional layers, three fully-connected layers are used to map 3D features to a lower dimensional space of 3D hand joint locations. In the first two fully-connected layers, we apply dropout layers with dropout rate 0.5 in order to prevent the neural network from overfitting [120].

4.3.1.2 3D Deep Dense Network

Deep convolutional networks with shortcut connections have shown powerful learning ability for image recognition [108, 117], since the shortcut connection can alleviate the gradient vanishing problem. We apply the deep dense network architecture [117] in our proposed 3D CNN-based hand pose estimation method. As shown in Figure 4.3b, we design a 3D deep dense network containing 28 convolutional layers and 3 fully-connected layers. Following the architecture in [117], we apply bottleneck layers in each dense block to reduce the number of model parameters. Experiments in Section 4.6.2.5 will show that the 3D deep dense network can achieve more accurate estimation result compared with that of the 3D shallow plain network.

4.3.2 Loss Function for 3D Hand Pose Regression

Our method estimates K hand joint locations in 3D space, which represent the 3D hand pose. Let the K objective hand joint locations be $\Phi = \{\phi_k\}_{k=1}^K \in \Lambda$, here Λ is the $3 \times K$ dimensional hand joint space. We denote a training sample as (X_n, Φ_n) , where X_n is the depth image, Φ_n is corresponding joint locations in the camera's coordinate system, $n = 1, \dots, N$. The depth image X_n is converted to the volumetric representation V_n as described in Section 4.2. The center of V_n is located at the center of the AABB generated from the 3D hand points. Thus, the ground truth Φ_n should be transformed into coordinates in the volume's coordinate system and normalized between 0 and 1. We denote the transformed and normalized joint locations as $\mathbf{Y}_n = \{y_{nk}\}_{k=1}^K \in \Lambda$. The 3D location $y_{nk} \in \mathbb{R}^3$ of the k -th hand joint is transformed and normalized as:

$$y_{nk} = (\phi_{nk} - c_n) / (M \cdot l_{voxel}) + 0.5, \quad (4.4)$$

where c_n is the center of the 3D volume. The original ground truth location ϕ_{nk} is transformed into the volume's coordinate system by subtracting the center c_n , and is normalized between -0.5 and 0.5 by dividing the volume's edge length l_{voxel} (we assume that all joints are within the 3D volume). To make the coordinates of y_{nk} be between 0 and 1, we add 0.5 in this formula.

Since the degree of freedom (DOF) of 3D hand joints is usually lower than the dimension of hand joint locations that is $3 \times K$, our designed 3D CNN explicitly enforces the hand configuration constraints on the estimation of hand joint locations, thus can alleviate infeasible hand pose estimations. To learn the hand pose priors, we perform PCA on the transformed and normalized joint locations $\{\mathbf{Y}_n\}_{n=1}^N$ in the training dataset, which is similar to the method in [94]. The coefficients of the principal components are $\alpha_n = \mathbf{E}^T \cdot (\mathbf{Y}_n - \mathbf{u})$, where α_n contains F coefficients, $F < 3 \times K$; $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_F]$ are the principal components; $\mathbf{u}$ is the mean vector of $\mathbf{Y}$.

During the training stage, we minimize the following objective function using the SGD algorithm:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_{n=1}^N \|\boldsymbol{\alpha}_n - \mathcal{F}(V_n, \mathbf{w})\|^2, \quad (4.5)$$

where $\mathcal{F}(\cdot)$ denotes the forward propagation of 3D CNN, $\mathbf{w}$ denotes the weights in the 3D CNN. During testing, given the input volumetric representation V , the transformed and normalized hand joint locations estimated by the 3D CNN is $\hat{\mathbf{Y}} = \mathbf{E} \cdot \mathcal{F}(V, \mathbf{w}^*) + \mathbf{u}$.

4.3.3 3D Data Augmentation

Another challenge of 3D hand pose estimation is that hand pose has large variations in global orientations and hand sizes. In order to make the 3D CNN model robust to different orientations and sizes and improve its generalization ability, we perform 3D data augmentation on the training data for both pose regression and surface completion networks. Different from existing 2D image data augmentation, our method directly rotate and stretch the hand points in 3D space.

We first stretch the point cloud along x , y , z axes of the camera's coordinate system with stretch factors s_x , s_y and s_z , respectively. Then, the point cloud is rotated around x , y , z axes of the camera's coordinate system with rotation angles θ_x , θ_y and θ_z , respectively. For a 3D point $\mathbf{p}$, after stretching and rotating, the point $\mathbf{p}$ is transformed into $\mathbf{p}'$:

$$\begin{aligned} \mathbf{p}' &= \mathcal{R} \cdot \mathcal{S} \cdot \mathbf{p} \\ \mathcal{R} &= \mathcal{R}_x(\theta_x) \cdot \mathcal{R}_y(\theta_y) \cdot \mathcal{R}_z(\theta_z) \\ \mathcal{S} &= \text{Diag}(s_x, s_y, s_z), \end{aligned} \quad (4.6)$$

where $\mathcal{R}_x$, $\mathcal{R}_y$ and $\mathcal{R}_z$ are 3×3 rotation matrices around x , y , z axes, respectively; $\text{Diag}(s_x, s_y, s_z)$ is a 3×3 diagonal matrix whose diagonal entries starting in the upper left corner are s_x , s_y and s_z . Figure 4.4 shows an example of 3D data augmentation. 3D stretching and rotation are performed on the hand point cloud

Figure 4.4: An example of 3D data augmentation. **Top-left:** original point cloud, ground truth and TSDF volume. **Bottom-left:** point cloud, ground truth and TSDF volume after 3D stretching. **Top-right:** point cloud, ground truth and TSDF volume after 3D rotation. **Bottom-right:** point cloud, ground truth and TSDF volume after 3D stretching and rotation. For illustration purpose, we only draw the projective D-TSDF volume on z direction.

and corresponding ground truth joint locations. TSDF volumes are then generated from the transformed point cloud.

In this work, an augmented training set is generated by randomly stretching and rotating original training samples. The stretch factors s_x and s_y are chosen log-uniformly at random from the interval $[1/1.5, 1.5]$. Since it is the relative size rather than the absolute size that affects the TSDF volume, we can set the stretch factor s_z as 1. In the camera’s coordinate system, the point cloud generated from the depth image is not only in-plane rotated around z axis, but also out-of-plane rotated around x , y axes, as shown in Figure 4.4. The out-of-plane rotated point cloud is different from the real point cloud due to the incompleteness of the point cloud. We assume that with small out-of-plane rotation angles θ_x and θ_y , the TSDF volume generated from the out-of-plane rotated point cloud can be approximately the same as the TSDF volume generated from the real point cloud when the camera’s viewpoint is at the rotated angle, and we choose θ_x and θ_y uniformly at random from the interval $[-30^\circ, 30^\circ]$. The in-plane rotation angle θ_z is chosen uniformly at random from the interval $[-180^\circ, 180^\circ]$. During the training stage, both the original training set and the augmented training set are used for

training. Experiments in Section 4.6.2.3 will show the effectiveness of the 3D data augmentation.

4.4 Hand Surface Completion

One of challenges in 3D hand pose estimation is that the input depth image only captures partial hand surface and suffers from the self-occlusion problem. To tackle this problem, we leverage the complete hand surface as intermediate supervision for learning 3D hand pose from depth images. More specifically, we estimate the complete hand surface using a data-driven approach and take advantage of the estimated complete hand surface for 3D hand pose estimation. As shown in Figure 4.5, we apply the 3D U-Net [121] architecture for estimating the complete hand surface from the captured partial hand surface. Similar to [122], we use the unsigned truncated distance function (TDF) as the network output, since it is not necessary to differentiate known and unknown space for the distance function of the complete hand surface. The ground truth of the complete hand surface is generated by using the model fitting method [2] with a 3D hand model. When training the 3D U-Net, we minimize the smooth L_1 loss defined in [123] using the ADAM optimizer [124]. After generating the TDF volume of the complete hand surface, we concatenate it with the original projective D-TSDF volumes and feed them into a 3D DenseNet for 3D hand pose estimation. The 3D U-Net and the 3D DenseNet are pre-trained separately, and are fine tuned in an end-to-end manner. Experiments in Section 4.6.2.6 will show that, with the intermediate supervision of hand surface completion, the accuracy of 3D hand pose estimation can be further improved.

Figure 4.5: The framework for hand surface completion and 3D hand pose estimation. 3D U-Net is applied to estimate the TDF volume of complete hand surface points from the projective D-TSDF volumes corresponding to the captured partial hand surface points. The projective D-TSDF volumes concatenated with the estimated TDF volume are fed into the 3D CNN for 3D hand pose estimation. ‘C’ denotes the concatenation operation.

4.5 Visualizing 3D CNN

In order to analyze 3D features extracted by the 3D CNN, we visualize the input patterns that produce given activations in the 3D feature volumes by adopting the

Figure 4.6: Visualization of patterns learned in a fully trained 3D CNN model. For each layer, we show reconstructed patterns in the 1st row, and corresponding receptive fields indicated by black boxes in the 2nd row. These patterns are reconstructed by using the guided backpropagation method proposed in [125]. For patterns of L3, we only draw voxels of which absolute values are larger than a threshold. Voxels with large values are shown in bright color, and voxels with small values are shown in dark color. Neurons in the 1st convolutional layer (L1) can capture local structures, such as corners and edges; neurons in the 2nd convolutional layer (L2) can capture structures of hand part, such as fingers; neurons in the 3rd convolutional layer (L3) can capture global structures of hand.

guided backpropagation method proposed in [125] which is a modification of the deconvolution based visualizing approach proposed in [126].

In Figure 4.6, we visualize some 3D patterns learned in a 3D shallow plain network which is fully trained on the MSRA hand pose dataset [1]. We take three input volumes with different hand poses as examples in Figure 4.6. In order to reconstruct 3D patterns, 3D feature volumes generated from 3D convolutional layers are projected down to the input voxel space by using the guided backpropagation method. For each convolutional layer of each hand pose, we choose four feature volumes as examples in Figure 4.6. To show which parts of the input volume cause high activations in the 3D feature volume, we crop the reconstructed 3D pattern volume inside the receptive field corresponding to the highest activation in the 3D feature volume. We also show the relative location of the receptive field in the input 3D volume. The receptive field sizes of the 1st convolutional layer (L1), the 2nd convolutional layer (L2) and the 3rd convolutional layer (L3) in the 3D shallow plain network are 5, 10 and 20, respectively.

As can be seen in Figure 4.6, from low layer (L1) to high layer (L3), neurons can capture patterns from local to global. Neurons in L1 capture low-level local geometry structures, such as the corners (e.g., L1 column 3, 8, 12) and edges (e.g., L1 column 1, 5, 7). Neurons in L2 capture mid-level shape structures of the hand, such as the hand finger tips (e.g., L2 column 5, 6, 8, 10) and palm edges (e.g., L2 column 2, 4, 7, 11). Neurons in L3 capture high-level global structures of the hand. It is interesting to note that, in patterns of L3, the contrast in regions of hand joints is obvious, which indicates that the learned high layer feature volumes focus on regions of hand joints. The hierarchical nature of 3D features is consistent with that in the 2D CNNs as observed in [126].

4.6 Experiments

We evaluate our proposed method on three public hand pose datasets: MSRA2015 hand pose dataset [1], NYU hand pose dataset [2] and ICVL hand pose dataset [3]. We also conduct corss-dataset experiments on MSRA2014 hand pose ataset [66].

Three metrics are employed to evaluate the hand pose estimation performance in our experiments. The first metric is the per-joint mean error distance over all test frames. The second metric is the proportion of good frames in which the worst joint error is below a threshold [111], which is a strict measure. The third metric is the proportion of joints within an error threshold [8].

4.6.1 Implementation Details

We train and evaluate our proposed 3D CNN models for 3D hand pose estimation on a computer with two Intel Core i7 5930K 3.50GHz CPUs, 64GB of RAM and an Nvidia GeForce GTX 1080 GPU having 8GB of GPU memory. The 3D CNN models are implemented within the PyTorch framework. For network training parameters, we choose the batch size as 16, the momentum as 0.9 and the weight decay as 0.0005. The learning rate is set as 0.01 for the 3D regression network, and is divided by 10 after 50 epochs. For the 3D U-Net, the learning rate is set as 0.001. The training is stopped after 60 epochs to prevent overfitting. We apply the same hyper-parameters for all experiments on all datasets. All the weights of 3D convolutional layers in 3D CNNs are randomly initialized using the method proposed in [127].

Figure 4.7: Self-comparison of different volume resolutions and different volume types with/without data augmentation on MSRA2015 dataset [1]. **Left:** the impact of different volume resolutions on the proportion of good frames. **Middle:** the impact of different volume types and data augmentation on the proportion of good frames. **Right:** the impact of different volume types and data augmentation on the per-joint mean error distance (R:root, T:tip).

4.6.2 Self-comparison

4.6.2.1 Choice of Volume Resolution

To evaluate the impact of different volume resolutions, we experiment with projective D-TSDF volumes with different resolution values: 16, 32 and 64 using the 3D plain networks with direct regression. Since training the network with $64 \times 64 \times 64$ volume resolution is very time consuming, we only train and test these three networks with different volume resolutions on a small subset of the MSRA2015 dataset without data augmentation in this experiment. As shown in Figure 4.7 (left), the estimation accuracy of $16 \times 16 \times 16$ resolution is slightly inferior to those with $32 \times 32 \times 32$ and $64 \times 64 \times 64$ resolutions. The estimation accuracy of the latter two resolutions is almost the same. However, computing TSDF volume with $64 \times 64 \times 64$ resolution is more time consuming and memory intensive. Thus, the volume resolution $32 \times 32 \times 32$ is most suitable for hand pose estimation, and we use this volume resolution in the following experiments. This experiment also shows that our method is robust to relatively low volume resolution, since the estimation accuracy does not decrease a lot when the resolution value is 16.

4.6.2.2 Choice of Volume Types

We evaluate the impact of different volume types on the estimation accuracy on MSRA2015 dataset using the 3D plain network without data augmentation. As can be seen in Figure 4.7 (middle and right), among occupancy grid, accurate TSDF, projective TSDF and projective D-TSDF, the projective D-TSDF performs best. It is worth noting that the performance of occupancy grid is comparable with those of accurate TSDF and projective TSDF, which indicates that the 3D CNNs can learn effective 3D features from the occupancy grid, although the occupancy grid cannot differentiate voxels before and behind the observed surface. But the projective D-TSDF, which encodes more information on three directions, outperforms the occupancy grid. For the real-time performance, the average computation time to generate occupancy grid, accurate TSDF, projective TSDF and projective D-TSDF on the same GPU are 1.4ms, 30.2ms, 1.9ms and 2.9ms, respectively. Thus, considering both the estimation accuracy and the real-time performance, the projective D-TSDF is overall best. In the following experiments, we apply the projective D-TSDF with $32 \times 32 \times 32$ volume resolution as the network input and apply 3D data augmentation for training.

4.6.2.3 Evaluation of Data Augmentation

We compare the method without using data augmentation and the methods using 2D/3D data augmentation. For 2D data augmentation, we randomly rotate and stretch the 3D hand point cloud in the 2D image plane. As shown in Figure 4.7 (middle and right), when using the same input volume, the method using 2D data augmentation outperforms the method without using data augmentation. When using 3D data augmentation in the training stage, the estimation accuracy is further improved. It is worth noting that, although the 3D rotated point cloud is not exactly the same as the real point cloud, the network trained with 3D data augmentation can still achieve better performance than the network trained with 2D data augmentation, which indicates that the network can benefit from the 3D augmented data.

4.6.2.4 2D CNNs Versus 3D CNNs

We compare the performance of the 3D CNN-based methods and the 2D CNN-based methods in this experiment. For 2D CNN-based methods, we segment the hand from the depth image and resize it to a 96×96 image while keeping the aspect ratio. The outputs of 2D CNNs are 2D image locations and depth values of hand joints, which are converted to 3D locations using camera parameters. For fair comparison, we experiment with the 2D deep dense network and 3D deep dense network having similar network architecture and comparable number of parameters. The 2D deep dense network has 5 dense blocks, 29 convolutional layers and 3 full-connected layers with 65M parameters; and the 3D deep dense network has 4 dense blocks, 28 convolutional layers and 3 full-connected layers with 50M parameters. In addition, we perform 2D data augmentation when training the 2D deep dense network, and the number of training samples is the same as that used in training the 3D deep dense network. As shown in Figure 4.8, the 3D deep dense network consistently performs better than the 2D deep dense network on all the three hand pose datasets, which indicates that the 3D CNNs can better utilize the depth information and provide more accurate estimation.

Figure 4.8: Self-comparison of 2D/3D CNNs with different network architectures for 3D hand pose estimation on MSRA2015 [1], NYU [2] and ICVL [3] hand pose datasets. The mean error distances over all joints of different methods are shown in the legends.

4.6.2.5 Shallow Network Versus Deep Network

We compare the estimation accuracy of the 3D shallow plain network and the 3D deep networks. For 3D deep networks, apart from the deep dense network, we also experiment with the deep residual network [108] which has 4 residual blocks, 9 convolutional layers and 3 full-connected layers. As can be seen in Figure 4.8, the performance of 3D deep dense network is better than that of 3D deep residual network, and the performance of 3D deep residual network is better than that of 3D shallow plain network on all the three hand pose datasets. Although the 3D deep networks have more convolutional layers, the convolutional layers in the 3D shallow plain network have more output channels. Thus, the forward propagation time of these three networks is comparable, which is 3.5ms for 3D shallow plain network, 3.4ms for 3D deep residual network, and 4.5ms for 3D deep dense network. We use the 3D deep dense network in the following experiments.

4.6.2.6 Evaluation of Hand Surface Completion

We evaluate the impact of hand surface completion on the accuracy of 3D hand pose estimation. Some examples of the estimated complete hand surface as the intermediate results of our method are shown in Figure 4.9. We extract the hand surface from the 32^3 volume of distance function using Matlab’s *isosurface* function [122]. As can be seen from Figure 4.9, our method is able to generate complete hand surface from the input partial hand surface. Compared with the ground truth, our estimation is more blurry and loses some details of the complete hand surface. The average L_1 loss of estimated TDF volume against the ground truth TDF volume of the complete surface is 0.112 in voxel space of which the truncation distance is 2.5. The average errors of 3D hand pose estimation using hand surface completion and without using hand surface completion are presented in Table 4.1. As can be seen, no matter using the 3D deep residual network or the 3D deep dense network, the surface completion step can further improve the estimation accuracy. In addition, in order to evaluate the importance of the complete hand surface on hand pose estimation, we use the ground truth of the complete hand surface combined with the original partial hand surface as network input to estimate the 3D hand pose. As presented in the last row of Table 4.1, the networks can achieve much smaller

Figure 4.9: Examples of hand surface completion with our method on NYU hand pose dataset [2]. The hand surfaces are extracted from the 32^3 volumes of distance function.

Table 4.1: Impact of Hand Surface Completion on 3D Hand Pose Estimation using 3D ResNet and 3D DenseNet. The Average Estimation Errors on NYU Dataset [2] are Listed in This Table.

	3D ResNet	3D DenseNet
w/o Surface Completion	12.1mm	10.6mm
/w Surface Completion	11.7mm	10.2mm
/w Ground Truth of Complete Surface	11.0mm	8.3mm

estimation errors when using the ground truth of the complete hand surface, which shows the potential of our method to achieve smaller pose estimation error if the complete hand surface could be estimated more accurately.

4.6.3 Comparison with State-of-the-art

4.6.3.1 Comparison on MSRA Dataset

On MSRA2015 dataset, we compare our 3D CNN-based hand pose estimation method with seven state-of-the-art methods: the hierarchical regression method [1], the joint matrix factorization and completion (JMFC) method [76], the local surface normals (LSN) based method [83], the crossing nets using deep generative models [89], the improved 2D CNN with hand pose prior and refinement (DeepPrior++) [92], the region ensemble network (REN) [128], the pose guided structured

Figure 4.10: Comparison with state-of-the-art methods [1, 76, 83, 89, 92, 93, 128] and our multi-view CNNs-based method (Chapter 3) on MSRA2015 dataset [1]. **Left:** the proportion of good frames over different error thresholds. **Middle & right:** the mean error distance over different yaw and pitch viewpoint angles with respect to the camera frame.

REN (Pose-REN) [93], and the our multi-view CNN-based method proposed in Chapter 3. Note that since the hierarchical regression method [1] has been shown superior to the methods in [4, 61], we indirectly compare our method with [4, 61].

As shown in Figure 4.10, our 3D CNN-based method outperforms state-of-the-art methods on the MSRA2015 dataset. The proportion of good frames over different error thresholds is shown in Figure 4.10 (left). Our method achieves the best performance when the error threshold is larger than 10mm. For example, when the error threshold is 30mm, the proportion of good frames of our method is about 6%, 10%, 20%, 22%, 28%, 33% and 39% higher than those of the methods in [93], [92], Chapter 3, [89], [83] (finger jointly regression), [1] and [76], respectively. When the error threshold is 5mm, the proportion of good frames of our method is slightly worse than those of the methods in [1] and [76]. This may be caused by the relatively large edge length of the voxel, which is 5.5mm on average when the volume resolution is $32 \times 32 \times 32$. In Figure 4.10 (middle and right), we compare the mean error distance over different yaw and pitch viewpoint angles with the methods in [1, 93, 128] and Chapter 3. As can be seen, the mean errors over different viewpoint angles of our method are about 7mm, 4.5mm, 1.8mm and 0.8mm smaller than those of the methods in [1], Chapter 3, [128] and [93], respectively. Our method exhibits less variance to the pitch viewpoint angle changes with a smaller standard deviation (0.58mm) than those of the methods in [1] (0.79mm), Chapter 3 (0.64mm), [128] (0.82mm) and [93] (0.63mm).

Figure 4.11: Comparison with state-of-the-art methods [2, 5, 88, 89, 92–96, 129] and our multi-view CNNs-based method (Chapter 3) on NYU dataset [2]. **Left:** the proportion of good frames over different error thresholds. **Middle:** the proportion of joints within different error thresholds. **Right:** the per-joint mean error distance (R:root, T:tip).

4.6.3.2 Comparison on NYU Dataset

On NYU hand pose dataset, we first compare our 3D CNN-based hand pose estimation method with nine state-of-the-art methods: the 2D CNN-based heat-map regression method [2], the 2D CNN-based direct regression method with a pose prior (DeepPrior) [94], the 2D CNN-based method using a feedback loop [5], the 2D CNN-based hand model parameter regression method [95], the deep feature based matrix completion method (DeepHand) [96], the crossing nets using deep generative models [89], the Lie-X method applying the Lie group theory [129], the DeepPrior++ method [92], and the Pose-REN method [93]. For the 2D CNN-based heat-map regression method [2], we estimate the 2D joint locations from heat-maps and convert them to 3D locations using corresponding depth values. As shown in Figure 4.11 (left), our method outperforms these nine methods over all the error thresholds. For example, the proportion of good frames of our method is about 10% more than that of the Pose-REN method [93] when the error threshold is between 20mm and 50mm.

In order to make a fair comparison with the spatial attention network based hierarchical hybrid method in [88], we evaluate the proportion of joints within different error thresholds on the subset of 11 hand joints following the experiment in [88] (removing palm joints except the root joint of thumb). As shown in Figure 4.11 (middle), our method is superior to the methods in [5, 88, 92–94] over all

Figure 4.12: Comparison with state-of-the-art methods [1, 3, 83, 89, 92, 93, 95, 128] on ICVL dataset [3]. **Left:** the proportion of good frames over different error thresholds. **Right:** the per-joint mean error distance (R:root, T:tip).

the error thresholds. For example, the proportion of joints within error threshold 20mm of our method is about 20% more than that of the method in [88].

We also compare the mean error distance of our method with those of the methods in [5, 92, 93, 95]. As shown in Figure 4.11 (right), our method achieves the smallest mean error distance on most joints, and the mean error distance over all joints of our method is about 5.5mm, 6.5mm, 2mm and 1.5mm smaller than those of methods in [5], [95], [92] and [93], receptively.

4.6.3.3 Comparison on ICVL Dataset

On ICVL hand pose dataset, we compare our 3D CNN-based hand pose estimation method with eight state-of-the-art methods: the latent regression forest (LRF) [3], the hierarchical regression method [1], the hand model parameter based method [95], the LSN method [83], the crossing nets using deep generative models [89], the REN method [90], the DeepPrior++ method [92], and the Pose-REN method [93]. As shown in Figure 4.12 (left), our method outperforms these eight methods over most of the error thresholds on this dataset.

We also compare the mean error distance of our method with those of the methods in [83, 93, 94, 128]. As shown in Figure 4.12 (right), our method achieves the smallest mean error distance on most joints. The mean error distance over all joints of our method is 6.7mm, while those of methods in [94], [83], [128], and [93] are 9.3mm, 8.2mm, 7.3mm and 6.8mm, receptively.

Table 4.2: Average Estimation Errors (in *mm*) of 6 Subjects for 7 Methods Tested on the Dataset Released in [66].

Subject	1	2	3	4	5	6	Avg
FORTH	35.4	19.8	27.3	26.3	16.6	46.2	28.6
PSO	29.3	14.8	40.2	17.3	16.2	24.3	23.6
ICP	29.9	20.7	30.8	23.9	18.5	32.8	26.1
ICP-PSO	10.1	24.1	13.0	12.8	11.9	20.0	15.3
MVCNN	30.1	19.7	24.3	19.9	21.8	20.7	22.8
3D DenseNet	14.2	11.4	11.4	10.8	10.9	11.6	11.7
ICP-PSO*	8.6	7.4	9.8	10.4	7.8	11.7	9.2

4.6.4 Cross-dataset Experiment

To evaluate the generalization ability of our 3D CNN-based hand pose estimation method, we perform a cross-dataset experiment, in which the 3D CNN model is trained on the whole MSRA2015 dataset [1] and is evaluated on the whole MSRA2014 dataset [66]. According to the evaluation metric in [66], we calculate the mean error distances for the wrist and the five fingertips. As shown in Table 4.2, we compare our 3D CNN-based method with model based tracking methods reported in [66], which are FORTH [27], PSO [66], ICP [113], ICP-PSO [66] and ICP-PSO* (ICP-PSO with finger-based initialization) [66], as well as our multi-view CNN-based method proposed in Chapter 3. As can be seen, our method achieves the second best result on three subjects and on the average error over all subjects.

It is worth noting that the model-based methods in [66] require a carefully calibrated hand model for each subject. However, our method does not use the calibrated hand model and thus is more flexible for different subjects. In such situation, our method still outperforms the FORTH, PSO, ICP and ICP-PSO methods, as shown in Table 4.2. Our method performs a little bit worse than the ICP-PSO* method which uses the ground truth to initialize the first frame. But our method does not use any ground truth of the testing data and is performed on cross-dataset which is more challenging. Thus, the overall second best result achieved by our method in this cross-dataset experiment indicates that our 3D CNN-based method has good generalization ability.

Figure 4.13: Qualitative results for MSRA2015, NYU and ICVL datasets. We compare our method based on the 3D deep dense network (in the 2nd column for each dataset) with the method based on the 2D deep dense network (in the 1st column for each dataset). The ground truth hand joint locations are presented in the last column for each dataset. We show hand joint locations and bones with the point cloud. Different hand joints and bones are visualized using different colors. This figure is best viewed in color.

4.6.5 Qualitative Results

Some qualitative results for MSRA2015, NYU and ICVL datasets are shown in Figure 4.13. We compare our 3D deep dense network with the baseline method of 2D deep dense network as described in Section 4.6.2.4. As can be seen in Figure 4.13, when using the same residual network architecture, our 3D CNN-based method can better utilize the depth information and provide more accurate estimation than the 2D CNN-based method.

We also conduct qualitative comparison with the Intel RealSense SDK [118]. We train the 3D CNN model on the whole MSRA2015 dataset and apply this pre-trained 3D CNN model to perform real-time hand pose estimation with the Intel RealSense SR300 Depth Camera in real scenarios. Qualitative results of our 3D CNN-based method and the Intel RealSense SDK [118] are shown in Figure 4.14. As can be seen, the Intel RealSense SDK does not accommodate complex hand poses as accurately as our method. For example, in the 4th column of Figure 4.14, when the little finger is occluded by the ring finger, the Intel RealSense SDK makes wrong estimation of ring finger joints' locations; in the 6th column of Figure 4.14,

Figure 4.14: Qualitative results for testing in real scenarios using Intel RealSense SR300 Depth Camera. For each hand pose, the 1st row shows the RGB image of the hand; the 2nd row shows the depth image of the hand; the 3rd row shows the hand pose estimated by the Intel RealSense SDK [118]; the 4th row shows the hand pose estimated by our 3D CNN-based method. We show hand joint locations and bones with the point cloud. Joints of thumb, index, middle, ring and little fingers are visualized in green, blue, cyan, yellow and pink, receptively. This figure is best viewed in color.

the Intel RealSense SDK confuses the index finger with the middle finger; in the 2nd, 8th and 9th columns of Figure 4.14, the Intel RealSense SDK confuses the ring finger with the little finger. By contrast, benefiting from the 3D CNN which can better exploit the 3D information, our method is able to correctly estimate the hand poses in these cases. More comparisons are presented in our demo video, available online.¹

4.6.6 Runtime and Model Size

The runtime of our 3D CNN-based hand pose estimation method is 7.9ms on average on the computer described in Section 4.6.1. The process of generating the projective D-TSDF volumes with $32 \times 32 \times 32$ volume resolution on GPU takes 2.9ms on average. The process of 3D deep dense network forward propagation running on GPU takes 4.5ms on average. The process of reconstructing 3D coordinates of hand joints in the 3D volume from PCA coefficients output by the 3D CNN and transforming them to 3D locations in the camera’s coordinate system takes 0.5ms on average on CPU. Thus, our method is capable of running in

¹<https://youtu.be/xdMebIYt2g8>

real-time at over 126 frames per second (fps). When adopting the hand surface completion step in our method, it takes 7.6ms on average for the 3D U-Net and the 3D deep dense network forward propagation. Thus, the runtime of the method using hand surface completion is 11.0ms, and the frame rate is 91 fps.

For model size, our 3D deep dense network model takes about 192MB, the 3D U-Net for hand surface completion takes about 200MB, while the multi-view CNNs proposed in Chapter 3 take about 1.2GB. The network parameters are stored in 32 bit float.

4.7 Applications

In this section, we develop two virtual reality applications using our proposed 3D hand pose estimation method, as shown in Figure 4.15.

The first application is virtual object manipulation. As shown in Figure 4.15a, the user wears an Oculus Rift headset [130] which provides virtual reality environment for the user. An Intel RealSense SR300 Depth Camera is put in front of the user to capture the depth images. By estimating the 3D hand pose from the depth images, our system enables user to manipulate virtual objects with the bare hand in the virtual reality environment. The second application is virtual 3D painting. As shown in Figure 4.15b, the user can use the index fingertip to paint in the virtual reality environment, and can also use bare hand to select color and rotate the 3D painting in the virtual reality environment.

These virtual reality applications demonstrate the potential of our 3D hand pose estimation method in real-world scenarios. More details of these applications are presented in our demo video, available online.²

²<https://youtu.be/aJAoN-utL9Q>

Figure 4.15: Applications of our proposed 3D CNN-based hand pose estimation method (a) User uses bare hand to rotate and move the virtual object in the virtual reality environment. (b) User uses bare hand to do virtual 3D painting in the virtual reality environment.

4.8 Summary

In this chapter, we present a novel 3D CNN-based hand pose estimation method. By adopting the projective D-TSDF, we encode the hand depth image as a 3D volumetric representation which is then fed into the 3D CNN. We show that the 3D CNN mapping the 3D volumes to 3D joint locations in a single pass is easy to be trained in an end-to-end manner. The 3D deep dense network can further improve the learning ability for 3D hand pose estimation. To make the 3D CNN robust to various hand sizes and global orientations, we perform 3D data augmentation on the training data. To tackle the self-occlusion problem, we leverage

the complete hand surface as intermediate supervision for learning 3D hand pose. Experimental results indicate that our proposed 3D CNN-based approach achieves superior performance for 3D hand pose estimation on three challenging hand pose datasets and is able to run in real-time with good generalization ability.

Chapter 5

Point Set-based Holistic Regression for Hand Pose Estimation

5.1 Overview

In Section 4, we encode the hand depth images as 3D volumes and apply a 3D CNN for inferring 3D hand pose. However, the time and space complexities of the 3D CNN grow cubically with the resolution of the input 3D volume [131]. Consequently, the 3D volume adopted in Section 4 is limited to a low resolution (e.g., 32^3), which may lose useful details of the hand. Furthermore, due to the sparsity of 3D point cloud, most of the voxels in the 3D volume are usually not occupied by any points, which not only wastes computations of 3D convolutions, but also distracts the neural networks from learning effective kernels to capture meaningful features of hand shapes. The approach in Section 4 transforms the sparse point cloud to a dense volumetric representation to enable effective 3D convolution. But this transformation changes the nature of the data and makes the data unnecessarily voluminous.

To tackle these problems, motivated by the recent works of PointNet [132, 133] that perform 3D object classification and segmentation on point sets directly, we

Figure 5.1: Overview of our proposed Hand PointNet-based method with holistic regression for 3D hand pose estimation in single depth images. We normalize the 3D point cloud in an oriented bounding box (OBB) to make the network input robust to global hand rotation. The 3D coordinates of sampled and normalized points attached with estimated surface normals are fed into a hierarchical PointNet [133], which is trained in an end-to-end manner, to extract hand features and regress 3D joint locations. The fingertip refinement PointNet can further improve the estimation accuracy of fingertip locations.

aim at learning 3D hand articulations directly from the 3D point cloud instead of rasterizing the 3D points into 3D voxels. It is worth noting that the depth image in essence is represented by a set of unordered 3D points on the visible surface of the hand, which is essentially 2.5D data and is not directly suitable to be processed by 2D or 3D convolutions. Compared with the multi-view CNNs-based method proposed in Chapter 3 and 3D CNN-based method proposed in Chapter 4, the point set-based approach does not need to project the point cloud into multiple 2D images or transform the sparse point cloud into dense 3D volumes, thus can better utilize the original point cloud in an effective and efficient way.

In this chapter, we propose a point set-based holistic hand joints regression method for 3D hand pose estimation in single depth images, as illustrated in Figure 5.1. Specifically, the segmented hand depth image is first converted to a set of 3D points; the 3D point cloud of the hand is downsampled and normalized in an oriented bounding box to make our method robust to various hand orientations. The hierarchical PointNet [133] takes the 3D coordinates of normalized points attached with the estimated surface normals as the input, and holistically regresses a low dimensional representation of the 3D hand joint locations which are then recovered in the camera coordinate system. The fingertip locations are further refined by a basic PointNet which takes the neighboring points of the estimated fingertip location as input. The contributions of this chapter are summarized as follows:

- We propose to estimate 3D hand joint locations directly from 3D point cloud based on the network architecture of PointNet [132, 133]. Compared with 2D CNN-based methods [2, 90–92] and multi-view CNNs-based method proposed in Chapter 3, our method can better exploit the 3D spatial information in the depth image; compared with 3D CNN-based method proposed in Chapter 4, our method can effectively leverage more information in the depth image to capture more details of the hand with fewer number of network parameters.
- In order to make our method robust to variations in hand global orientations, we propose to normalize the sampled 3D points in an oriented bounding box without applying any additional network to transform the hand point cloud. The normalized point clouds with more consistent global orientations make the PointNet easier to learn 3D hand articulations.
- We propose to refine the fingertip locations with a basic PointNet that takes the neighboring points of the estimated fingertip location as input to regress the refined fingertip location. The refinement network can further exploit finer details in the original point cloud and regress more accurate fingertip locations.

5.2 PointNet Revisited

Basic PointNet: PointNet [132] is a type of neural network that directly takes a set of points as the input and is able to extract discriminative features of the point cloud. As shown in Figure 5.2, each input point $\mathbf{x}_i \in \mathbb{R}^D$ ($i = 1, \dots, N$) is mapped into a C -dim feature vector through multi-layer perceptron (MLP) networks, of which the weights across different points are shared. A vector max operator is applied to aggregate N point feature vectors into a global feature vector that is invariant to permutations of input points. Finally, the C -dim global feature vector is mapped into an F -dim output vector using MLP networks. It has been proved in [132] that PointNet has the ability to approximate arbitrary continuous set functions, given enough neurons in the network.

Hierarchical PointNet: The main limitation of the basic PointNet is that it cannot capture local structures of the point cloud in a hierarchical way. To address this problem, Qi et al. [133] proposed a hierarchical PointNet which has better generalization ability due to its hierarchical feature extraction architecture. In this work, we exploit the hierarchical PointNet for 3D hand pose estimation. As shown in Figure 3.1, the hierarchical PointNet consists of L point set abstraction levels. At the l -th level ($l = 1, \dots, L - 1$), N_l points are selected as centroids of local regions; the k -nearest neighbors of the centroid point are grouped as a local region; a basic PointNet with shared weights across different local regions is applied to extract a C_l -dim feature of each local region; N_l centroid points with d -dim coordinates and C_l -dim features are fed into the next level. At the last level, a global point cloud feature is abstracted from the whole input points of this level by using a basic PointNet.

5.3 Point Cloud Normalization

The hand depth image is first converted to a set of 3D points using the depth camera’s intrinsic parameters. The 3D point set is then downsampled to N points.

One challenge of 3D hand pose estimation is the large variation in global orientation of the hand. To make our method robust to various hand orientations, we normalize the hand point cloud. The objective for hand point cloud normalization is to transform the original hand point cloud into a canonical coordinate system

Figure 5.2: Basic architecture of PointNet. The network directly takes N points as the input. Each D -dim input point is mapped into a C -dim feature through MLP. Per-point features are aggregated into a global feature by max-pooling. The global feature is mapped into an F -dim output vector.

in which the global orientations of the transformed hand point clouds are as consistent as possible. This normalization step ensures that our method is robust to variations in hand global orientations.

If the output is invariant to the rotation of input point cloud, such as the outputs in semantic labeling tasks [132], we can add a spatial transformation network [132, 134] into the PointNet to predict the transformation matrix, as proposed in [132]. However, in our problem, the output 3D hand pose depends on the orientation of the input hand point cloud, which makes the network difficult to be trained in an end-to-end manner. Some solutions could be designing two parallel networks to estimate the canonical hand pose and the hand orientation at the same time [45] or adding a spatial de-transformation network before the output layer to remap the estimated pose to the original coordinate system (C.S.) [135]. But these solutions will increase the complexity of the network and make the training more difficult.

In this work, we propose a simple yet effective method to normalize the 3D hand point cloud in OBB, instead of applying any additional networks to estimate the

Figure 5.3: Examples of OBB-based point cloud normalization. The 1st and 3rd columns present the original point clouds in the camera C.S. with various hand orientations. The 2nd and 4th columns present the sampled and rotated point clouds in the OBB C.S., of which the hand orientations are more consistent. We color-code all point clouds to show the relative z coordinate value.

hand global orientation or transform the hand point cloud. OBB is a tightly fitting bounding box of the input point cloud [105]. The orientation of OBB is determined by performing principal component analysis (PCA) on the 3D coordinates of input points. The x, y, z axes of the OBB C.S. are aligned with the eigenvectors of input points' covariance matrix, which correspond to eigenvalues from largest to smallest, respectively. The original points in camera C.S. are first transformed into OBB C.S., then these points are shifted to have zero mean and scaled to a unit size:

$$\begin{aligned}\mathbf{p}^{obb} &= (\mathbf{R}_{obb}^{cam})^T \cdot \mathbf{p}^{cam}, \\ \mathbf{p}^{nor} &= (\mathbf{p}^{obb} - \bar{\mathbf{p}}^{obb}) / L_{obb},\end{aligned}\tag{5.1}$$

where $\mathbf{R}_{obb}^{cam}$ is the rotation matrix of the OBB in camera C.S.; $\mathbf{p}^{cam}$ and $\mathbf{p}^{obb}$ are 3D coordinates of point $\mathbf{p}$ in camera C.S. and OBB C.S., respectively; $\bar{\mathbf{p}}^{obb}$ is the centroid of point cloud $\{\mathbf{p}_i^{obb}\}_{i=1}^N$; L_{obb} is the maximum edge length of OBB; $\mathbf{p}^{nor}$ is the normalized 3D coordinate of point $\mathbf{p}$ in the normalized OBB C.S.

During training, the ground truth 3D joint locations in camera C.S. also apply the transformation in Equation 5.1 to obtain the 3D joint locations in the normalized OBB C.S. During testing, the estimated 3D joint locations in the normalized OBB C.S. $\hat{\phi}_m^{nor}$ are transformed back to those in camera C.S. $\hat{\phi}_m^{cam}$ ($m = 1, \dots, M$):

$$\hat{\phi}_m^{cam} = \mathbf{R}_{obb}^{cam} \cdot (L_{obb} \cdot \hat{\phi}_m^{nor} + \bar{\mathbf{p}}^{obb}).\tag{5.2}$$

Figure 5.3 presents some examples of the original hand point clouds and the corresponding normalized hand point clouds. As can be seen, although the orientations of original hand point clouds in camera C.S. have large variations, the orientations of normalized hand point cloud in OBB C.S. are more consistent. Experiments are conducted in Section 5.6.2 to show that our proposed OBB-based point cloud normalization method can improve the performance of both the basic PointNet and the hierarchical PointNet.

5.4 Hand Pose Regression Network

We design a 3D hand pose regression network which can be trained in an end-to-end manner. The input of the hand pose regression network is a set of normalized points $\mathbf{X}^{nor} = \{\mathbf{x}_i^{nor}\}_{i=1}^N = \{(\mathbf{p}_i^{nor}, \mathbf{n}_i^{nor})\}_{i=1}^N$, where $\mathbf{p}_i^{nor}$ is the 3D coordinate of the normalized point, and $\mathbf{n}_i^{nor}$ is the corresponding 3D surface normal which is approximated by performing PCA on the nearest neighboring points of the query point in the sampled point cloud to fit a local plane [138]. These N points are then fed into a hierarchical PointNet [133], as shown in Figure 5.4, which has three point set abstraction levels. The first two levels group input points into $N_1 = 512$ and $N_2 = 128$ local regions, respectively. Each local region contains $k = 64$ points. These two levels extract $C_1 = 128$ and $C_2 = 256$ dimensional features for each local region, respectively. The last level extracts a 1024-dim global feature vector which is mapped to an F -dim output vector by three fully-connected layers. Each MLP network in Figure 5.4 is composed of several fully connected layers. All fully connected layers are followed by batch normalization and ReLU except for the last layer of the last MLP network. We do not use dropout layer in our implementation.

Since the degree of freedom of human hand is usually lower than the dimension of 3D hand joint locations ($3 \times M$), the PointNet is designed to output an F -dim ($F < 3 \times M$) representation of hand pose to enforce hand pose constraint and alleviate infeasible hand pose estimations, which is similar to [94]. In the training phase, given T training samples with the normalized point cloud and the corresponding ground truth 3D joint locations $\{(\mathbf{X}_t^{nor}, \Phi_t^{nor})\}_{t=1}^T$, we minimize the following objective function:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_{t=1}^T \|\alpha_t - \mathcal{F}(\mathbf{X}_t^{nor}, \mathbf{w})\|^2 + \lambda \|\mathbf{w}\|^2 \quad (5.3)$$

where $\mathbf{w}$ denotes network parameters; $\mathcal{F}$ represents the hand pose regression PointNet; λ is the regularization strength; α_t is an F -dim projection of Φ_t^{nor} . By performing PCA on the ground truth 3D joint locations in the training dataset, we can obtain $\alpha_t = \mathbf{E}^T \cdot (\Phi_t^{nor} - \mathbf{u})$, where $\mathbf{E}$ denotes the principal components, and

Figure 5.4: Architecture of the proposed point set-based holistic hand pose regression network. Numbers in parentheses of MLP networks are layer sizes. In our implementation, $N = 1024$, $N_1 = 512$, $N_2 = 128$, $k = 64$, $D = 6$, $d = 3$, $C_1 = 128$, $C_2 = 256$, $C_3 = 1024$. We set the dimension of output vector as $F = 2 \times M < 3 \times M$, where M is the number of hand joints, $3 \times M$ is the dimension of hand joint locations.

$\mathbf{u}$ is the empirical mean. During testing, the estimated 3D joint locations are reconstructed from the network outputs:

$$\hat{\Phi}^{nor} = \mathbf{E} \cdot \mathcal{F}(\mathbf{X}^{nor}, \mathbf{w}^*) + \mathbf{u}. \quad (5.4)$$

In Figure 5.5, we visualize the sensitivity of points in different regions to different filters at three set abstraction levels in a well trained three-level hierarchical PointNet. The corresponding input is the example shown in Figure 3.1. We normalize per-point features in the same region extracted by the same filter between 0 and 1. Large normalized feature value means that the corresponding point is sensitive to the filter. As can be seen in Figure 5.5, from low level to high level, the size of receptive field in Euclidean space becomes larger and larger. At the first two levels, filters extract features from local regions, and different filters exaggerate different local structures in the local region. At the highest level, filters extract

Figure 5.5: Visualization of the point sensitivity in different regions to different filters at three set abstraction levels. At each of the first two levels, each column corresponds to the same local region, and each row corresponds to the same filter. For illustration purpose, we only show the sensitivity of points in three local regions to two filters at each of the first two levels; At the third level, we show the sensitivity of points to six filters. Points with high sensitivity are shown in red, and points with low sensitivity are shown in blue. Points that do not belong to the local region are shown in gray color.

features from the whole input points of this level, and different filters exaggerate different parts of the hand, such as different fingers, hand palm, *etc.* With such a hierarchical architecture, the network can capture structures of the hand from local to global.

5.5 Fingertip Refinement Network

To further improve the estimation accuracy of fingertip locations, we design a fingertip refinement network which takes K nearest neighboring points of the estimated fingertip location as input and outputs the refined 3D location of the fingertip, as shown in Figure 5.6. Note that the K nearest neighboring points of the estimated fingertip location are searched in the original input point cloud of hand, and K is set as 256 in our implementation. We refine fingertip locations for two reasons: first, the estimation error of fingertip locations is usually relatively large compared to other joints, as shown in experimental results in Section 5.6.2.3; second, the fingertip location of straightened finger is usually easy to be refined, since the K nearest neighboring points of the fingertip will not change a lot even if the estimated location deviates from the ground truth location to some extent when K is relatively large.

Figure 5.6: Illustration of fingertip refinement. K nearest neighboring points of the estimated fingertip location are found in the original hand point cloud with upper limit of point number. These points are normalized and fed into a basic PointNet that outputs the refined fingertip location. Numbers in parentheses of MLP networks are layer sizes.

Since we only refine fingertips for straightened fingers, we first check each finger is bent or straightened by calculating joint angles using the joint locations. For the straightened finger, we find the K nearest neighboring points of the fingertip location in the original point cloud with upper limit of point number to ensure real-time performance. The K nearest neighboring points are then normalized in OBB, which is similar to the method in Section 5.3. A basic PointNet takes these normalized points as input and outputs the refined fingertip 3D location. During the training stage, we use the ground truth joint locations to calculate joint angles; for the fingertip location used in the nearest neighbor search, we add a 3D random offset within a radius of $r = 15mm$ to the ground truth fingertip location in order to make the fingertip refinement network more robust to inaccurate fingertip estimations. During the testing stage, we use joint locations estimated by the hand pose regression network for calculating joint angles and searching nearest neighboring points.

5.6 Experiments

We evaluate our proposed method on three public hand pose datasets: NYU [2], MSRA [1] and ICVL [3].

5.6.1 Implementation Details

For training PointNets, we use Adam [136] optimizer with initial learning rate 0.001, batch size 32 and regularization strength 0.0005. The learning rate is divided by 10 after 50 epochs. The training process is stopped after 60 epochs.

For fingertip refinement, we set the number of nearest neighboring points K as 256. Considering the real-time performance, we downsample the original point cloud to $N' = 6000$ points with random sampling when the number of points in the original point cloud exceeds the upper limit N' , and we apply Kd-tree algorithm [137] in the efficient nearest neighbor search.

In addition, the 3D surface normal is approximated by performing PCA on the nearest neighboring points of the query point in the sampled point cloud to fit a local plane [138]. The number of nearest neighboring points is set as 30. For the hierarchical PointNet, followed by [133], we use farthest point sampling method to sample centroids of local regions and ball query to group points. The radius for ball query is set as 0.1 at the first level and 0.2 at the second level. The output dimension of the hand pose regression network F is set as $2 \times M$, which is 2/3 of the dimension of the hand joint locations.

All experiments are conducted on a workstation with two Intel Core i7 5930K, 64GB of RAM and an Nvidia GTX1080 GPU. The deep neural networks are implemented within the PyTorch framework.

Figure 5.7: Self-comparison of different methods on NYU dataset [2]. **Left:** the impact of different numbers of sampled points on the proportion of good frames. **Middle & Right:** the impact of different PointNet architectures and normalization methods on the proportion of good frames as well as on the per-joint mean error distance (R: root, T: tip). The overall mean error distances are shown in parentheses.

5.6.2 Self-comparisons

5.6.2.1 Choice of the Number of Sampled Points

We first evaluate the influence of the number of sampled points N . As shown in Figure 5.7 (left), we experiment with three different numbers of sampled points by using a basic PointNet for hand pose regression on NYU dataset. When $N = 512$, the estimation accuracy is slightly lower than the other two results with larger number of sampled points. But when $N = 1024$ and $N = 2048$, the estimation accuracy is almost the same. Balancing between estimation accuracy and real-time performance, we choose $N = 1024$ and apply this number of sampled points in the following experiments. This experiment also shows that our method is robust to a small amount of sampled points, since the performance does not drop a lot when $N = 512$.

5.6.2.2 Basic PointNet Versus Hierarchical PointNet

We evaluate the impact of different PointNet architectures and normalization methods on NYU dataset without fingertip refinement. Note that, for the normalization method without using OBB, we shift the point cloud center to zero and scale the points into a unit sphere without rotating the point cloud. As presented in Figure 5.7 (middle and right), with the same PointNet architecture, the estimation

Figure 5.8: The impact of fingertip refinement on fingertips’ mean error distances and the overall mean error distance on NYU [2], MSRA [1] and ICVL [3] datasets (T: tip).

accuracy of OBB-based point cloud normalization method is superior to that of the normalization method without using OBB by a large margin, which indicates that our proposed OBB-based point cloud normalization method is quite effective since it can normalize point cloud with more consistent orientations and make the network easier to learn the hand articulations. In addition, when adopting the same normalization method, the hierarchical PointNet performs better than the basic PointNet on the estimation accuracy, which shows that the hierarchical network architecture [133] is also effective in our hand joints regression task.

5.6.2.3 Evaluation of Fingertip Refinement

In addition, we study the influence of fingertip refinement. As can be seen in Figure 5.8, after fingertip refinement, the average estimation errors of most fingertips as well as the overall mean error decrease on all the three datasets, which indicates that our proposed fingertip refinement method can further improve the estimation accuracy of fingertip locations.

Figure 5.9: Comparison with state-of-the-art methods on NYU [2] (left), MSRA [1] (middle) and ICVL [3] (right) datasets. The proportions of good frames and the overall mean error distances (in parentheses) are presented in this figure.

5.6.3 Comparisons with State-of-the-art Methods

We compare our point set-based holistic hand joints regression method with 16 state-of-the-art methods: latent random forest (LRF) [3], hierarchical regression with random forest (Hierarchical) [1], collaborative filtering [76], 2D CNN for heat-map regression (Heat-map) [2], 2D CNN with prior and refinement (DeepPrior) [94], 2D CNN with feedback loop (Feedback) [5], 2D CNN for hand model regression (DeepModel) [95], matrix completion with deep feature (DeepHand) [96], local surface normal based random forest (LSN) [83], multi-view CNNs proposed in Chapter 3, 2D CNN using Lie group theory (Lie-X) [129], Crossing Nets [89], 3D CNN proposed in Chapter 4, region ensemble network (REN) [90], improved DeepPrior (DeepPrior++) [92] and 2D CNN with hallucinating heat distribution (Hallucination) [91]. The proportion of good frames over different error thresholds and the per-joint mean error distances of different methods on NYU [2], MSRA [1] and ICVL [3] datasets are presented in Figure 5.9 and Figure 5.10, respectively.

As can be seen in Figure 5.9, our method outperforms these state-of-the-art methods over most of the error thresholds. On NYU dataset [2], when the error threshold is between 15mm and 20mm, the proportions of good frames of our method is about 15% better than REN [90] and DeepPrior++ [92] methods. On MSRA dataset [1], when the error threshold is between 15mm and 20mm, the proportions of good frames of our method is about 10% better than 3D CNN proposed in Chapter 4 and DeepPrior++ [92] methods. On ICVL dataset [3], when the error threshold is 15mm, the proportions of good frames of our method is

Figure 5.10: Comparison with state-of-the-art methods on NYU [2] (left), MSRA [1] (middle) and ICVL [3] (right) datasets. The per-joint mean error distances and the overall mean error distances are presented in this figure (R: root, T: tip).

Figure 5.11: Qualitative results for NYU [2] (left), MSRA [1] (middle) and ICVL [3] (right) datasets. We compare the point set-based holistic regression method (in the second row) with the 3D CNN-based method in Section 4 (in the first row). The ground truth hand joint locations are presented in the last row. We show hand joint locations on depth images. Different hand joints and bones are visualized with different colors.

about 15% better than REN [90] method and 10% better than DeepPrior++ [92] method. For the mean error distances shown in Figure 5.10, our method also outperforms state-of-the-art methods on most of the hand joints and achieves the smallest overall mean error distances on all the three datasets.

5.6.4 Qualitative Results

Some qualitative results for NYU [2], MSRA [1] and ICVL [3] datasets are shown in Figure 5.11. As can be seen, compared with the 3D CNN-based method, the point set-based method can capture complex hand structures and estimate more accurate 3D hand joint locations.

5.6.5 Runtime and Model Size

During the testing stage, the depth image is first converted to a set of 3D points, from which 1024 points are randomly sampled. Since we use the hierarchical PointNet [133], the farthest point sampling (FPS) algorithm is applied to evenly sample 512 and 256 points from the 1024 points for the first two set abstraction levels. This step including random sampling and FPS takes 1.7ms for one frame in average. The next step is surface normal approximation for the 1024 points, which takes 6.5ms for one frame in average. These two steps are implemented using C++ with the Point Cloud Library (PCL). The 3D points attached with surface normals are then fed into the hand pose regression network to estimate 3D hand joint locations, which takes 9.2ms for one frame in average on a single Nvidia GTX1080 GPU. For fingertip refinement, neighboring points search and surface normal approximation take 2.8ms for one frame in average, and the fingertip refinement network forward propagation takes 0.3ms for one frame in average on a single Nvidia GTX1080 GPU. Therefore, the total runtime of our method is 20.5ms for one test frame in average, and the average frame rate is about 48fps.

The number of parameters in the hand pose regression network is about 2.3×10^6 (2.3M). The number of parameters in the fingertip refinement network is about 1.4×10^5 (142K). In total, there are about 2.5×10^6 (2.5M) parameters in these two networks. These parameters are stored in 32 bit float and the total size of parameters is 9.8MB.

5.7 Summary

In this chapter, we present a novel approach that directly takes point cloud as network input to holistically regress 3D hand joint locations. To handle variations of hand global orientations, we normalize the hand point clouds in OBB with more consistent global orientations. The normalized point cloud is fed into a hierarchical PointNet for hand pose regression. Fingertip locations are further refined by a basic PointNet. Experiments on three public hand pose datasets show that our method achieves superior performance for 3D hand pose estimation in real-time.

Chapter 6

Point Set-based Point-wise Regression for Hand Pose Estimation

6.1 Overview

In Section 5, we propose to holistically regress 3D coordinates of hand joints from point sets with the help of PointNet [132, 133]. However, the direct mapping from input representation to 3D hand pose is highly non-linear and difficult to learn, which makes these holistical regression methods difficult to achieve high accuracy [139]. An alternative way is to generate a set of heat-maps representing the probability distributions of joint locations on 2D image plane [2], which has been successfully applied in 2D human pose estimation [140, 141]. However, it is non-trivial to lift 2D heat-maps to 3D joint locations [142–144]. One straightforward solution is to generate volumetric heat-maps using 3D CNNs, but this solution is computationally inefficient. Wan et al. [97] recently propose a dense pixel-wise estimation method. Apart from generating 2D heat-maps, this method estimates 3D offsets of hand joints for each pixel of the 2D image. However, this method suffers from two limitations. First, as it regresses pixel-wise 3D estimations from 2D images, the proposed method may not fully exploit the 3D spatial information

Figure 6.1: Overview of our proposed point-wise regression method for 3D hand pose estimation from single depth images. We propose to directly take N sampled and normalized 3D hand points as network input and output a set of heat-maps as well as unit vector fields on the input point cloud, reflecting the closeness and directions from input points to J hand joints, respectively. From the network outputs, we can infer point-wise offsets to hand joints and estimate the 3D hand pose with post-processing. We apply the hierarchical PointNet [133] with two-stacked network architecture which feeds the output of one module as input to the next. For illustration purpose, we only visualize the heat-map, unit vector field and offset field of one hand joint. ‘C.S.’ stands for coordinate system; ‘MLP’ stands for multi-layer perceptron network.

in depth images. Second, generating 3D estimations for background pixels of the 2D image may distract the deep neural network from learning effective features in the hand region.

To tackle these problems, we aim at regressing point-wise estimations directly from 3D point cloud, since the depth image is intrinsically composed of a set of 3D points on the visible object surface. We take advantages of PointNet [132, 133] to learn features directly from 3D point cloud. In this chapter, we define *point-wise estimations* as the offsets from 3D points in the point cloud to hand joint locations.

Compared with the dense pixel-wise estimation method [97], our method can better utilize the 3D spatial information in the depth image in an efficient way and concentrate on learning effective features of the hand point cloud in a natural way, since both the input and the output of our network directly take the form of hand point cloud. In addition, this point-to-point regression scheme also allows us to expand the single hierarchical PointNet module [133] to a stacked network architecture as in [141] to further improve the estimation accuracy.

Compared with the holistic regression method proposed in Section 5 that holistically regresses 3D hand pose parameters from point cloud using a holistic regression PointNet, our method estimates the *point-wise* closeness and offset directions to hand joints from the input point cloud using a stacked point-to-point regression PointNet, which is able to capture local evidence for estimating accurate 3D hand pose.

As illustrated in Figure 6.1, we propose a point-to-point regression method for 3D hand pose estimation in single depth images. Hand is first segmented from the depth image and is converted to a set of 3D points. The downsampled and normalized 3D points are then fed into a hierarchical PointNet [133] with two-stacked network architecture. The outputs of the network are heat-maps and unit vector fields on the 3D point cloud, reflecting the closeness and directions from 3D points to the target hand joints, respectively. Point-wise offsets to hand joints are inferred from the network outputs and are used to vote for 3D hand joint locations. With post-processing steps to alleviate some limitations, the estimation accuracy is further improved. The contributions of this chapter are summarized as follows:

- We propose to directly take the 3D point cloud as network input and generate heat-maps as well as unit vector fields on the input point cloud, which reflect the per-point closeness and directions to hand joints, respectively. With such a point-to-point regression network, our method is able to better utilize the 3D spatial information in the depth image and capture local structure of the 3D point cloud for accurate 3D hand pose estimation.
- We propose to apply the stacked network architecture [141] to the hierarchical PointNet [133] for point-to-point regression, which is the first stacked PointNet architecture, to our best knowledge. Similar to [141], the stacked PointNet architecture, feeding the output of one module as input into the next, allows repeated bottom-up and top-down inference on 3D point cloud and is able to boost the estimation accuracy in our experiments.

- We analyze the limitations of our point-to-point regression method and propose to use results of holistic regression method as the alternative when the divergence among candidate estimations of point-to-point regression method is too large. Experiments show that the holistic regression method is complementary with the point-to-point regression method and their combination can further improve the estimation accuracy.

6.2 Point Set-based Representation of 3D Hand Pose

Our proposed method aims at estimating 3D hand pose from single depth images. The input is a depth image containing a hand, and the output is a set of 3D hand joint locations $\Phi^{cam} = \{\phi_j^{cam}\}_{j=1}^J \in \mathbf{\Lambda}$ in the camera Coordinate System (C.S.), where J is the number of hand joints, $\mathbf{\Lambda}$ is the $3 \times J$ dimensional hand joint space.

The hand depth image is first converted to a set of 3D points using the depth camera’s intrinsic parameters. The 3D point set is then downsampled to N points. Same as the point cloud normalization step in Section 5.3, we create an oriented bounding box (OBB) from the 3D point cloud and transform the 3D points into the OBB C.S. to make our method robust to various hand orientations. The coordinates of 3D points are normalized between -0.5 and 0.5 by subtracting the centroid of point cloud and dividing by L_{obb} , which is the maximum edge length of OBB. We denote the downsampled and normalized 3D point set in OBB C.S. as $\mathcal{P}^{obb} = \{\mathbf{p}_i^{obb}\}_{i=1}^N$. In our implementation, we set the number of sampled points N as 1024. Since we process the point set in OBB C.S., we will omit the superscript ‘obb’ in symbols of points and joint locations in the following sections for simplicity.

Most existing CNN-based methods for 3D hand pose estimation directly regress 3D coordinates of hand joints [5, 89, 90, 93] or hand pose parameters [91, 92, 94, 95]. In contrast to direct regression approaches that require to learn a highly non-linear mapping, our method aims at generating point-wise estimations of hand joint locations from the point cloud, which is able to better utilize the local evidence. The point-wise estimations can be defined as the offsets from points to hand joint locations. However, estimating offsets for all points in the point set is unnecessary

Figure 6.2: An illustration of the ground truth of the point cloud based representation for 3D hand pose. We visualize the neighboring points, offset field, heat-map and unit vector field on 3D point cloud for the root joint of the thumb finger. For illustration propose, we enlarge the region of neighboring points of the hand joint location on the right of each complete point cloud.

and may make the per-point votes noisy. Thus, we only estimate offsets for the neighboring points of the hand joint, as shown in Figure 6.2. We define the element in the target offset fields $\mathbf{V}$ for point $\mathbf{p}_i$ ($i = 1, \dots, N$) and ground truth hand joint location ϕ_j^* ($j = 1, \dots, J$) as:

$$\mathbf{V}(\mathbf{p}_i, \phi_j^*) = \begin{cases} \phi_j^* - \mathbf{p}_i & \mathbf{p}_i \in \mathcal{P}_K(\phi_j^*) \text{ and } \|\phi_j^* - \mathbf{p}_i\| \leq r, \\ \mathbf{0} & \text{otherwise;} \end{cases} \quad (6.1)$$

where $\mathcal{P}_K(\phi_j^*)$ is a set of K nearest neighboring points (KNN) of the ground truth hand joint location ϕ_j^* in the point set $\mathcal{P}^{obb}$; r is the maximum radius of ball for nearest neighbor search; in our implementation, we set K as 64 and r as $80mm/L_{obb}$. We combine KNN with ball query for nearest neighbor search in order to guarantee that both the number of neighboring points and the scale of neighboring region are controllable.

However, it is difficult to train a neural network that directly generates the offset field due to the large variance of offsets. Similar to [97], we decompose the target offset fields $\mathbf{V}$ into heat-maps H reflecting per-point closeness to hand joint locations:

$$H(\mathbf{p}_i, \phi_j^*) = \begin{cases} 1 - \|\phi_j^* - \mathbf{p}_i\|/r & \mathbf{p}_i \in \mathcal{P}_K(\phi_j^*) \text{ and } \|\phi_j^* - \mathbf{p}_i\| \leq r, \\ 0 & \text{otherwise;} \end{cases} \quad (6.2)$$

and unit vector fields $\mathbf{U}$ reflecting per-point directions to hand joint locations:

$$\mathbf{U}(\mathbf{p}_i, \phi_j^*) = \begin{cases} (\phi_j^* - \mathbf{p}_i) / \|\phi_j^* - \mathbf{p}_i\| & \mathbf{p}_i \in \mathcal{P}_K(\phi_j^*) \text{ and } \|\phi_j^* - \mathbf{p}_i\| \leq r, \\ \mathbf{0} & \text{otherwise.} \end{cases} \quad (6.3)$$

Different from [97] that generates heat-maps and unit vector fields on 2D images, our proposed method generates heat-maps and unit vector fields on the 3D point cloud, as shown in Figure 6.2, which can better utilize the 3D spatial information in the depth image. In addition, generating heat-maps and unit vector fields on 2D images with large blank background regions may distract the neural network from learning effective features in the hand region. Although this problem can be alleviated by multiplying a binary hand mask in the loss function, our method is able to concentrate on learning effective features of the hand point cloud in a natural way without using any mask, since the output heat-maps and unit vector fields are represented on the hand point cloud.

6.3 Network Architecture of Point-wise Regression PointNet

In this work, we exploit the hierarchical PointNet [133] for learning heat-maps and unit vector fields on 3D point cloud. Different from the hierarchical PointNet for point set segmentation adopted in [133], our proposed point-to-point regression network has a two-stacked network architecture in order to better capture the 3D spatial information in the 3D point cloud.

We first describe the network architecture of a single hierarchical PointNet module. As illustrated in Figure 6.3, the input of the network is a set of d -dim coordinates with C_{in} -dim input features, *i.e.*, 3D surface normals that are approximated by fitting a local plane for the nearest neighbors of the query point in the point cloud ($d = 3$ and $C_{in} = 3$ in this work). Similar to the network architecture for set segmentation proposed in [133], a single module of our network extracts a

global feature vector from point cloud using three set abstraction levels and propagates the global feature to point features for original points using three feature propagation levels, as shown in Figure 6.3. In the feature propagation level, we use nearest neighbors of the interpolation point in N_l points to interpolate features for N_{l-1} points [133]. The interpolated C_l -dim features of N_{l-1} points are concatenated with the corresponding point features in the set abstraction level and are mapped to C_{l-1} -dim features using per-point MLP, of which the weights are shared across all the points ($l = 1, 2, 3; N_0 = N, C_0 = C_{out}, N_3 = 1$). The heat-map and the unit vector field are generated from the point features for the original point set using per-point MLP. In our implementation, we set $N = 1024$, $N_1 = 512$, $N_2 = 128$, $C_1 = 128$, $C_2 = 256$ and $C_{out} = 128$.

Inspired by the stacked hourglass networks for human pose estimation [141], we stack two hierarchical PointNet modules end-to-end to boost the performance of the network. The two hierarchical PointNet modules have the same network architecture and the same hyper-parameters, except for the hyper-parameter in the input layer. As shown in Figure 6.4, the output heat-map and unit vector field of the first module are concatenated with the input and output point features of the first module as the input into the second hierarchical PointNet module. For real-time consideration, we only stack two hierarchical PointNet modules.

Figure 6.3: An illustration of a single network module which is based on the hierarchical PointNet [133]. Here, ‘SA’ stands for point set abstraction layers; ‘FP’ stands for feature propagation layers; ‘MLP’ stands for multi-layer perceptron network. The dotted shortcuts denote skip links for feature concatenation.

Figure 6.4: An illustration of the two-stacked hierarchical PointNet architecture with intermediate supervision. The input feature dimension of the 2nd network module is $C_{in2} = C_{in1} + C_{out} + 4J$.

We apply intermediate supervision when training the two-stacked hierarchical PointNet. The loss function for each training sample is defined as:

$$\mathcal{L} = \sum_{t=1}^T \sum_{j=1}^J \sum_{i=1}^N \left[\left(\hat{H}_{ij}^{(t)} - H(\mathbf{p}_i, \phi_j^*) \right)^2 + \left\| \hat{\mathbf{U}}_{ij}^{(t)} - \mathbf{U}(\mathbf{p}_i, \phi_j^*) \right\|^2 \right], \quad (6.4)$$

where T is the number of stacked network modules, in this work $T = 2$; $\hat{H}_{ij}^{(t)}$ and $\hat{\mathbf{U}}_{ij}^{(t)}$ are elements in the heat-maps and unit vector fields estimated by the t -th network module, respectively; $H(\mathbf{p}_i, \phi_j^*)$ and $\mathbf{U}(\mathbf{p}_i, \phi_j^*)$ are elements in the ground truth heat-maps and ground truth unit vector fields defined in Eq. 6.2 and Eq. 6.3, respectively.

6.4 Hand Pose Inference

6.4.1 Hand Pose Inference from Point-wise Estimation

During testing, we infer the 3D hand pose from the heat-maps $\hat{H}$ and the unit vector fields $\hat{\mathbf{U}}$ estimated by the last hierarchical PointNet module. According to the definition of offset fields, heat-maps and unit vector fields in Eq. 6.1-6.3, we can infer the offset vector $\hat{\mathbf{V}}_{ij}$ from point $\mathbf{p}_i$ to joint $\hat{\phi}_j$ as:

$$\hat{\mathbf{V}}_{ij} = r \cdot (1 - \hat{H}_{ij}) \cdot \hat{\mathbf{U}}_{ij}. \quad (6.5)$$

According to Eq. 6.1, only the offset vectors for the neighboring points of the hand joint are used for hand pose inference, which can be found from the estimated heat-map reflecting the closeness of points to the hand joint. We denote the estimated heat-map for the j -th hand joint as $\hat{H}_j$ that is the j -th column of $\hat{H}$. We determine the neighboring points of the j -th hand joint as the points corresponding to the largest M values of the heat-map $\hat{H}_j$. The indices of these points in the point set are denoted as $\{i_m\}_{m=1}^M$. The hand joint location $\hat{\phi}_j$ can be simply inferred from the corresponding offset vectors $\hat{\mathbf{V}}_{i_m j}$ and 3D points $\mathbf{p}_{i_m}$ ($m = 1, \dots, M$) using weighted average:

$$\hat{\phi}_j = \sum_{m=1}^M w_m \left(\hat{\mathbf{V}}_{i_m j} + \mathbf{p}_{i_m} \right) / \sum_{m=1}^M w_m, \quad (6.6)$$

where w_m is the weight of the candidate estimation. In our implementation, we set the weight w_m as the corresponding heat-map value $\hat{H}_{i_m j}$, and set M as 25.

6.4.2 Post-processing

There are two issues in our point-to-point regression method. The first issue is that the estimation is unreliable when the divergence of the M candidate estimations are large in 3D space, as shown in Figure 6.5a. This is usually caused by missing depth data near the hand joint. The second issue is that there is no explicit constraint on the estimated 3D hand pose, although the neural network may learn joint constraints in the output heat-maps and unit vector fields.

To tackle the first issue, when the divergence of the M candidate estimations is larger than a threshold, we replace the estimation result with the result of the direct regression method that directly regresses 3D coordinates of hand joints, since the direct regression method does not have this issue. In order to save the inference time, instead of training a separate PointNet for direct hand pose regression, we add three fully-connected layers for direct hand pose regression to the pre-trained two-stacked hierarchical PointNet, as shown in Figure 6.5b. The three fully-connected layers are trained to directly regress the 3D coordinates of hand joints from the features extracted by the second hierarchical PointNet module. The divergence of the M candidate estimations is defined as the sum of standard deviations of x ,

Figure 6.5: (a) A failure case in which the candidate estimations of the middle fingertip can not converge to a small local region in 3D space due to missing depth data near the hand joint. The ground truth hand joint locations are plotted in this figure. (b) An illustration of the two-stacked hierarchical PointNet architecture in which we add three fully-connected layers to directly regress the 3D coordinates of hand joints from the global feature extracted by the second hierarchical PointNet module.

y and z coordinates of candidate estimations. In our implementation, we set the divergence threshold as $7.5mm/L_{obb}$. Experimental results in Section 6.5.2.3 will show that although only a small portion of the hand joint estimations requires to be replaced by the direct regression results, this replacement strategy can improve the estimation accuracy to some extent.

To tackle the second issue, we explicitly constrain the estimated 3D hand pose $\hat{\Phi}$ on a lower dimensional space learned by principal component analysis (PCA). By performing PCA on the ground truth 3D joint locations in the training dataset, we can obtain the principal components $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_H]$ ($H < 3J$) and the empirical mean $\mathbf{u}$. The constrained 3D hand pose can be calculated using the following formula:

$$\hat{\Phi}_{cons} = \mathbf{E} \cdot \mathbf{E}^T \cdot (\hat{\Phi} - \mathbf{u}) + \mathbf{u}. \quad (6.7)$$

In our implementation, we set the number of principle components H as 30. Experimental results in Section 6.5.2.3 will show that adding PCA constraint will improve the accuracy slightly, which shows that the neural network may have already learned joint constraints in the output heat-maps and unit vector fields.

Finally, the estimated 3D hand joint locations in the normalized OBB C.S. are transformed back to joint locations in the camera C.S. $\hat{\Phi}^{cam}$.

6.5 Experiments

We evaluate our proposed method on three public hand pose datasets: NYU dataset [2], ICVL dataset [3] and MSRA dataset [1].

6.5.1 Implementation Details

The deep neural network models are implemented within the PyTorch framework. When training the deep neural network models, we use Adam [14] optimizer with initial learning rate 0.001, batch size 32, momentum 0.5 and weight decay 0.0005. The learning rate is divided by 10 after 30 epochs. The training is stopped after 60 epochs to prevent overfitting.

6.5.2 Self-comparisons

6.5.2.1 Evaluation of Stacked Network Architecture

We first evaluate the impact of the stacked network architecture for hierarchical PointNet. As shown in Figure 6.6 (left and right), the two-stacked network evidently performs better than the single network module, which indicates the importance of the stacked network architecture on our point-wise regression method.

6.5.2.2 Evaluation of Different Network Outputs

We also evaluate the impact of different network outputs. In our method, we train the network to output heat-maps and unit vector fields for hand joints, then use them to recover the offset fields, as described in Section 6.2. In this experiment, we compare our method with a baseline method in which a network is trained to generate offset fields instead of unit vector fields. The network also outputs heat-maps which are only used to find neighboring points of hand joints. As shown in Figure 6.6 (left), when adopting the two-stacked network architecture, the network generating unit vector fields performs better than the network generating offset fields. This result shows that the network regressing unit vectors of offsets may be easier to learn than the network regressing offset vectors, since the variance of the offset vectors is larger than the unit vectors.

Figure 6.6: Self-comparison of different methods on NYU dataset [2]. **Left:** the impacts of the stacked network architecture and different network outputs on the proportion of good frames. **Middle:** the impacts of our point-wise regression method and post-processing methods on the proportion of good frames. We use two-stacked network for point-wise regression in this figure. **Right:** the impact of point-to-point regression method, stacked network architecture and post-processing methods on the per-joint mean error distance (R: root, T: tip). ‘P2P Reg.’ stands for point-to-point regression. The overall mean error distances are shown in parentheses.

6.5.2.3 Point-wise Regression Versus Holistic Regression

To evaluate our proposed point-wise regression method, we compare our method with the holistic regression method proposed in Section 5. In this experiment, we use a hierarchical PointNet [133] with three set abstraction levels and three fully-connected levels to holistically regress the 3D coordinates of hand joints. As shown in Figure 6.6 (middle), our point-wise regression method outperforms the holistic regression method when the error threshold is smaller than 45mm. But when the error threshold is larger than 45mm, our point-wise regression method performs worse than the holistic regression method. This may be caused by the large divergence of the candidate estimations in some results, as described in Section 6.4.2. By combining the point-wise method with the holistic regression method as described in Section 6.4.2, the estimation accuracy can be further improved, as shown in Figure 6.6 (middle). Furthermore, the performance of the combination method is superior to or on par with the holistic regression method over all the error thresholds. In this experiment, only 7.9% of joint locations estimated by point-wise regression method are replaced by the results of holistic regression method, which indicates that the estimation results are dominated by the point-wise regression method, and the holistic regression method is complementary with the point-wise

regression method. In addition, adding the PCA constraint can further improve the estimation accuracy slightly.

6.5.2.4 Evaluation of Hand Pose Inference

We further study the influence of the number of candidate estimations M used in Eq. 6.7 and the weighted average on the overall mean error distance. As shown in Table 6.1, the mean error distance is the smallest when the number of candidate estimations M is between 15 and 25. When M is larger than 25, the mean error distance will become larger. In addition, when M is smaller than 25, the weighted average will not improve the mean error distance. But when M becomes larger, the improvement of the weighted average on the mean error distance is more and more evident. Thus, the weighted average is able to make the estimation more robust to noisy candidate estimations. We set M as 25 and use weighted average with post-processing in the following experiments.

6.5.3 Comparisons with State-of-the-arts

We compare our proposed point-wise regression method with 16 state-of-the-art methods: latent random forest (LRF) [3], hierarchical regression with random forest (RDF, Hierarchical) [1], local surface normal based random forest (LSN) [83], collaborative filtering [76], 2D heat-map regression using 2D CNNs (Heat-map) [2], feedback loop based 2D CNNs (Feedback Loop) [5], hand model parameter regression using 2D CNNs (DeepModel) [95], Lie group based 2D CNNs (Lie-X) [129], improved holistic regression with a pose prior using 2D CNNs (DeepPrior++) [92], hallucinating heat distribution using 2D CNNs (Hallucination Heat) [91], multi-view CNNs proposed in Chapter 3, 3D CNNs proposed in Chapter 4, crossing nets using deep generative models (Crossing Nets) [89], region ensemble network

# Candidate Estimations M	5	15	25	35	45	55
Unweighted Average	9.50mm	9.47mm	9.48mm	9.57mm	9.70mm	9.84mm
Weighted Average	9.50mm	9.47mm	9.46mm	9.53mm	9.61mm	9.71mm

Table 6.1: The impacts of the number of candidate estimations M and weighted average on the overall mean error distance on NYU dataset [2].

(REN) [90], pose guided structured REN (Pose-REN) [93] and dense 3D regression using 2D CNNs (DenseReg) [97]. We evaluate the proportion of good frames over different error thresholds and the per-joint mean error distances as well as the overall mean error distance on NYU [2], ICVL [3] and MSRA [1] datasets, as presented in Figure 6.7 and Figure 6.8, respectively.

As can be seen in Figure 6.7 and Figure 6.8, our method can achieve superior performance on these three datasets. On NYU [2] and ICVL [3] datasets, our method outperforms other methods over almost all the error thresholds and achieves the smallest overall mean error distances on these two datasets. Specifically, on NYU dataset [2], when the error threshold is between 15mm and 20mm, the proportions of good frames of our method is about 15% better than DenseReg [97] and 20% better than Pose-REN [93]; on ICVL dataset [3], when the error threshold is between 10mm and 15mm, the proportions of good frames of our method is more than 10% better than DenseReg [97] and Pose-REN [93] methods. On MSRA dataset [1], our method outperforms other methods over almost all the error thresholds, except for the DenseReg [97] method. Although our method is about 10% better than DenseReg [97] when the error threshold is 10mm and the overall mean error distance of our method is only 0.5mm worse than that of DenseReg [97], our method is worse than the DenseReg [97] method when the error threshold is larger 15mm. As mentioned in [10] and shown in the last two columns of Figure 6.9, some of the 3D hand joint annotations in MSRA dataset [1] exhibit significant errors, which may make the evaluation on this dataset less meaningful and may limit the learning ability of our deep neural network.

6.5.4 Qualitative Results

We present some qualitative results for NYU [2], ICVL [3] and MSRA [1] datasets in Figure 6.9. As can be seen in this figure, compared with the holistic regression method using hierarchical PointNet [133], our proposed point-wise regression method can estimate more accurate 3D hand joint locations.

Figure 6.7: Comparison with state-of-the-art methods on NYU [2] (left), ICVL [3] (middle) and MSRA [1] (right) datasets. The proportions of good frames and the overall mean error distances (in parentheses) are presented in this figure.

Figure 6.8: Comparison with state-of-the-art methods on NYU [2] (left), ICVL [3] (middle) and MSRA [1] (right) datasets. The per-joint mean error distances and the overall mean error distances are presented in this figure (R: root, T: tip).

6.5.5 Runtime and Model Size

The runtime of our method is 23.9ms per frame in average, including 8.2ms for point sampling and surface normal calculation, 15.1ms for the two-stacked hierarchical PointNet forward propagation, 0.6ms for hand pose inference and post-processing. Thus, our method runs in real-time at about 41.8fps.

In addition, the model size of our network is 17.2MB, including 11.1MB for the point-wise regression network which is a two-stacked hierarchical PointNet and 6.1MB for the additional holistic regression module which consists of three fully-connected layers.

Figure 6.9: Qualitative results for NYU [2] (left), ICVL [3] (middle) and MSRA [1] (right) datasets. We compare our method (in the second row) with the PointNet-based holistic regression method (in the first row). The ground truth hand joint locations are presented in the last row. We show hand joint locations on depth images. Different hand joints and bones are visualized with different colors. This figure is best viewed in color.

6.6 Summary

In this chapter, we propose a novel approach that directly takes the 3D point cloud of hand as network input and outputs heat-maps as well as unit vector fields on the point cloud, reflecting the per-point closeness and directions to hand joints. We infer 3D hand joint locations from the estimated heat-maps and unit vector fields using weighted fusion. Similar to the stacked hourglass network [141], we apply the stacked network architecture for the hierarchical PointNet [133], which allows repeated bottom-up and top-down inference on point cloud and is able to further boost the performance. Our proposed point-wise regression method can also be easily combined with holistic regression method to achieve more robust performance. Experimental results on three challenging hand pose datasets show that our method achieves superior accuracy performance in real-time.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

In this thesis, we systematically study the methods for 3D hand pose estimation from single depth images. The conclusions of this thesis are summarized in the following two aspects.

Firstly, one part of this thesis is targeted at better utilizing the 3D spatial information of hand in the input depth image. To achieve this target, we represent the hand depth images as multi-view projections, 3D volumes, and 3D point cloud in this thesis. More specifically, in Chapter 3, we project the 3D point cloud generated from the depth image onto multiple views and use multi-view CNNs to estimate heat-maps on each projected image. In Chapter 4, we encode hand depth image as a 3D volumetric representation, and use 3D CNN to extract features from the 3D volumes. In Chapter 5 and Chapter 6, we directly use the point cloud generated from the depth image as the neural network input. Experimental results in these four chapters have shown that the neural network is able to estimate more accurate 3D hand pose by lifting 2D depth maps to 3D representations as the input.

Secondly, when considering the neural network outputs, this thesis investigate the detection-based method and regression-based method. In Chapter 3, we apply the detection-based method to generate heat-maps for multiple views, and fuse the multi-view heat-maps to get the final 3D hand pose estimation. In Chapter 6,

we generate heat-maps and vector fields on the 3D point set, from which we can infer the 3D hand pose. While in Chapter 4 and Chapter 5, we use the regression-based method to directly regress the 3D hand joint locations or a lower dimensional representation of the 3D hand pose. According to our experimental results, both of these two methods have their advantages. Detection-based method is robust, while regression-based method can learn hand joint constraints implicitly.

In general, in this thesis, several effective methods have been proposed with promising experimental results on public hand pose datasets. These methods should also be beneficial for other articulated pose estimation problems and 3D data understanding problems.

7.2 Future Works

Although our proposed 3D hand pose estimation methods have achieved state-of-the-art performance, there are still many research problems on hand pose analysis need to be resolved, e.g., hand pose estimation from hand-object or hand-hand interactions, 3D hand pose estimation from RGB images, high-level hand action understanding.

Hand pose estimation from hand-object or hand-hand interactions.

In this thesis, we assume that the hand in the depth image is not interacting with objects or other hands. However, hand-object interactions and hand-hand interactions are common in our daily life. It is meaningful for computers to understand hand pose from hand-object or hand-hand interactions. But the severe occlusion and more variations in hand-object or hand-hand interactions make this task more challenging than hand pose estimation without interactions. Some works have investigated this research problem [55, 145, 146]. Our proposed methods in this thesis could be applied in this problem in future work.

3D hand pose estimation from RGB images. Compared to depth cameras, color cameras are more commonly available, and are not constrained in indoor environments. Thus, 3D hand pose estimation from RGB images has broad application prospects. But 3D hand pose estimation from RGB images is more challenging than that from depth images, due to the severe ambiguities in RGB

images, large appearance variations of the background, and lack of explicit 3D information in RGB images.

High-level hand action understanding. Only estimating 3D hand pose is not enough for high-level human-computer interaction. It is promising to make computers understand high-level hand actions in our daily life. Specifically, future research could focus on hand action recognition and detection. The high-level 3D features extracted by our methods in this thesis could be adopted in these research problems.

Author's Publications

Journal Papers

1. Liuhao Ge, Hui Liang, Junsong Yuan and Daniel Thalmann. Real-time 3D Hand Pose Estimation with 3D Convolutional Neural Networks. *Accepted by IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*.
2. Liuhao Ge, Hui Liang, Junsong Yuan and Daniel Thalmann. Robust 3D Hand Pose Estimation from Single Depth Images using Multi-View CNNs. In *IEEE Transaction on Image Processing (T-IP)*, vol. 27, no. 9, pp. 4422-4436, 2018.
3. Hui Liang, Junsong Yuan, Jun Lee, Liuhao Ge, Daniel Thalmann. Hough Forest With Optimized Leaves for Global Hand Pose Estimation With Arbitrary Postures. In *IEEE Transactions on Cybernetics (T-CYB)*, vol. PP, no. 99, pp. 1-15, 2017.

Conference Papers

1. Liuhao Ge, Zhou Ren, and Junsong Yuan. Point-to-Point Regression PointNet for 3D Hand Pose Estimation. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.
2. Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3D Hand Pose Estimation from Monocular RGB Images. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

3. Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. Hand PointNet: 3D Hand Pose Estimation using Point Sets. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
4. 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
5. 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Bibliography

- [1] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, “Cascaded hand pose regression,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 824–832.
- [2] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, “Real-time continuous pose recovery of human hands using convolutional networks,” *ACM Trans. Graph.*, vol. 33, no. 5, pp. 169:1–169:10, 2014.
- [3] D. Tang, H. J. Chang, A. Tejani, and T. K. Kim, “Latent regression forest: Structured estimation of 3D articulated hand posture,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 3786–3793.
- [4] C. Xu and L. Cheng, “Efficient hand pose estimation from a single depth image,” in *Proc. Int. Conf. Comput. Vis.*, 2013, pp. 3456 – 3462.
- [5] M. Oberweger, P. Wohlhart, and V. Lepetit, “Training a feedback loop for hand pose estimation,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 3316–3324.
- [6] M. Oberweger, G. Riegler, P. Wohlhart, and V. Lepetit, “Efficiently creating 3D training data for fine hand pose estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 3593–3601.
- [7] H. Liang, J. Yuan, and D. Thalmann, “Resolving ambiguous hand pose predictions by exploiting part correlations,” *IEEE Trans. on Circuits and Syst. for Video Technol.*, vol. 25, no. 7, pp. 1125–1139, Jul. 2015.

- [8] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rheimann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi, “Accurate, robust, and flexible real-time hand tracking.” *Proc. 33rd Annual ACM Conf. Human Factors in Computing Systems*, April 2015, pp. 3633–3642.
- [9] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, S. Izadi, R. Banks, A. Fitzgibbon, and J. Shotton, “Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences,” *ACM Trans. Graph.*, vol. 35, no. 4, pp. 143:1–143:12, 2016.
- [10] M. Oberweger, G. Riegler, P. Wohlhart, and V. Lepetit, “Efficiently creating 3d training data for fine hand pose estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 4957–4965.
- [11] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim, “Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 2605–2613.
- [12] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Y. Chang, K. M. Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, J. Yuan, X. Chen, G. Wang, F. Yang, K. Akiyama, Y. Wu, Q. Wan, M. Madadi, S. Escalera, S. Li, D. Lee, I. Oikonomidis, A. Argyros, and T.-K. Kim, “Depth-based 3D hand pose estimation: From current achievements to future goals,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 2636 – 2645.
- [13] S. Han, B. Liu, R. Wang, Y. Ye, C. D. Twigg, and K. Kin, “Online optical marker-based hand tracking with deep labels,” *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–10, 2018.
- [14] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, “A review on vision-based full dof hand motion estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshops*, 2005, pp. 75–82.

- [15] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 3593–3601.
- [16] ———, "Robust 3d hand pose estimation from single depth images using multi-view cnns," *IEEE Trans. Image Processing*, vol. 27, no. 9, pp. 4422–4436, 2018.
- [17] ———, "3D convolutional neural networks for efficient and robust hand pose estimation from single depth images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 1991–2000.
- [18] ———, "Real-time 3d hand pose estimation with 3d convolutional neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–15, 2018.
- [19] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand pointnet: 3D hand pose estimation using point sets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 8417–8426.
- [20] L. Ge, Z. Ren, and J. Yuan, "Point-to-point regression pointnet for 3d hand pose estimation," in *Proc. European Conf. Comput. Vis.*, 2018.
- [21] D. L. Quam, "Gesture recognition with a dataglove," in *Proc. IEEE Conf. Aerospace and Electronics*, vol. 2, 1990, pp. 755–760.
- [22] C. Wang and D. J. Cannon, "A virtual end-effector pointing system in point-and-direct robotics for inspection of surface flaws using a neural network based skeleton transform," in *Proc. Int. Conf. Robotics and Automation*, vol. 3, 1993, pp. 784–789.
- [23] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: a review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 677–695, 1997.
- [24] Y. Wu and T. S. Huang, "Vision-based gesture recognition: A review," in *International Gesture Workshop*, 1999, pp. 103–115.

- [25] —, “Hand modeling, analysis and recognition,” *IEEE Signal Processing Magazine*, vol. 18, no. 3, pp. 51–60, 2001.
- [26] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, “Vision-based hand pose estimation: A review,” *Computer Vision and Image Understanding*, vol. 108, no. 1, pp. 52 – 73, 2007.
- [27] I. Oikonomidis, N. Kyriazis, and A. Argyros, “Efficient model-based 3D tracking of hand articulations using Kinect,” in *Proc. British Mach. Vis. Conf.*, 2011, pp. 101.1–101.11.
- [28] M. Kitagawa and B. Windsor, *MoCap for artists: workflow and techniques for motion capture*. Focal Press, 2012.
- [29] C.-S. Chua, H. Guan, and Y.-K. Ho, “Model-based 3d hand posture estimation from a single 2d image,” *Image and Vision Computing*, vol. 20, no. 3, pp. 191 – 202, 2002.
- [30] C. Theobalt, I. Albrecht, J. Haber, M. Magnor, and H.-P. Seidel, “Pitching a baseball: Tracking high-speed motion with multi-exposure images,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 540–547, 2004.
- [31] R. Y. Wang and J. Popović, “Real-time hand-tracking with a color glove,” *ACM Trans. Graph.*, vol. 28, no. 3, pp. 63:1–63:8, 2009.
- [32] W. Zhao, J. Chai, and Y.-Q. Xu, “Combining marker-based mocap and rgb-d camera for acquiring high-fidelity hand motion data,” in *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2012, pp. 33–42.
- [33] S. Alexanderson, C. O’Sullivan, and J. Beskow, “Robust online motion capture labeling of finger markers,” in *Proc. Int. Conf. Motion in Games*, 2016, pp. 7–13.
- [34] M. Schröder, J. Maycock, and M. Botsch, “Reduced marker layouts for optical motion capture of hands,” in *Proc. ACM SIGGRAPH Conf. Motion in Games*, 2015, pp. 7–16.

- [35] M. Schröder, T. Waltemate, J. Maycock, T. Röhlig, H. Ritter, and M. Botsch, “Design and evaluation of reduced marker layouts for hand motion capture,” *Computer Animation and Virtual Worlds*, 2017.
- [36] J. Maycock, T. Rohlig, M. Schroder, M. Botsch, and H. Ritter, “Fully automatic optical motion tracking using an inverse kinematics approach,” in *IEEE-RAS Int. Conf. Humanoid Robots*, 2015, pp. 461–466.
- [37] N. Wheatland, S. Jörg, and V. Zordan, “Automatic hand-over animation using principle component analysis,” in *Proc. Motion on Games*, 2013, pp. 175:197–175:202.
- [38] B. Stenger, P. R. Mendonça, and R. Cipolla, “Model-based 3d tracking of an articulated hand,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, vol. 2, 2001, pp. II–310.
- [39] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla, “Model-based hand tracking using a hierarchical bayesian filter,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1372–1384, 2006.
- [40] Y. Wu and T. S. Huang, “View-independent recognition of hand postures,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2000, pp. 88–94.
- [41] Y. Wu, J. Lin, and T. S. Huang, “Analyzing and capturing articulated hand motion in image sequences,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1910–1922, 2005.
- [42] V. Athitsos and S. Sclaroff, “Estimating 3d hand pose from a cluttered image,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, vol. 2, 2003, pp. II–432.
- [43] M. Bray, E. Koller-Meier, and L. Van Gool, “Smart particle filtering for 3d hand tracking,” in *Proc. Int. Conf. Automatic Face and Gesture Recog.*, 2004, pp. 675–680.

- [44] M. de La Gorce, D. J. Fleet, and N. Paragios, “Model-based 3D hand pose estimation from monocular video,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1793–1805, 2011.
- [45] C. Zimmermann and T. Brox, “Learning to estimate 3D hand pose from single RGB images,” in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 4903–4911.
- [46] P. Panteleris, I. Oikonomidis, and A. Argyros, “Using a single rgb frame for real time 3d hand pose estimation in the wild,” in *Proc. IEEE Winter Conf. Applications of Comput. Vis.*, 2017.
- [47] A. Spurr, J. Song, S. Park, and O. Hilliges, “Cross-modal deep variational hand pose estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 89–98.
- [48] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, “GANerated hands for real-time 3d hand tracking from monocular rgb,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 49–59.
- [49] U. Iqbal, P. Molchanov, T. Breuel, J. Gall, and J. Kautz, “Hand pose estimation via latent 2.5D heatmap regression,” in *Proc. European Conf. Comput. Vis.*, 2018.
- [50] Y. Cai, L. Ge, J. Cai, and J. Yuan, “Weakly-supervised 3d hand pose estimation from monocular rgb images.” in *Proc. European Conf. Comput. Vis.*, 2018.
- [51] S. Sridhar, A. Oulasvirta, and C. Theobalt, “Interactive markerless articulated hand motion tracking using rgb and depth data,” in *Proc. Int. Conf. Comput. Vis.*, 2013, pp. 2456–2463.
- [52] J. Lien, N. Gillian, M. E. Karagozler, P. Amihoud, C. Schwesig, E. Olson, H. Raja, and I. Poupyrev, “Soli: Ubiquitous gesture sensing with millimeter wave radar,” *ACM Trans. Graph.*, vol. 35, no. 4, pp. 142:1–142:19, 2016.

- [53] J. M. Rehg and T. Kanade, “Visual tracking of high dof articulated structures: an application to human hand tracking,” in *Proc. European Conf. Comput. Vis.*, 1994, pp. 35–46.
- [54] L. Ballan, A. Taneja, J. Gall, L. V. Gool, and M. Pollefeys, “Motion capture of hands in action using discriminative salient points,” in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 640–653.
- [55] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall, “Capturing hands in action using discriminative salient points and physics simulation,” *International Journal of Computer Vision*, vol. 118, no. 2, pp. 172–193, 2016.
- [56] H. Joo, T. Simon, and Y. Sheikh, “Total capture: A 3d deformation model for tracking faces, hands, and bodies,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 8320–8329.
- [57] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, “Panoptic studio: A massively multiview system for social motion capture,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 5474–5483.
- [58] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. S. Godisart, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, “Panoptic studio: A massively multiview system for social interaction capture,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.
- [59] L. Motion. (2018) Leap motion. [Online]. Available: <https://www.leapmotion.com>
- [60] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges, “Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum,” in *Proc. Annual Symposium on User Interface Software and Technology*. ACM, 2016, pp. 851–860.

- [61] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-time human pose recognition in parts from a single depth image,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 1297–1304.
- [62] M. Ye, Y. Shen, C. Du, Z. Pan, and R. Yang, “Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera.” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1517–1532, 2016.
- [63] H. Y. Jung, Y. Suh, G. Moon, and K. M. Lee, “A sequential approach to 3d human pose estimation: Separation of localization and identification of body joints,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 747–761.
- [64] J. S. Supancic III, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan, “Depth-based hand pose estimation: methods, data, and challenges,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 1868–1876.
- [65] A. Tagliasacchi, M. Schroeder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, “Robust articulated-ICP for real-time hand tracking,” *Computer Graphics Forum*, vol. 34, no. 5, pp. 101–114, Jul. 2015.
- [66] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, “Realtime and robust hand tracking from depth,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 1106–1113.
- [67] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, “Markerless and efficient 26-dof hand pose recovery,” in *Proc. Asian Conf. Comput. Vis.* Springer, 2010, pp. 744–757.
- [68] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt, “Fast and robust hand tracking using detection-guided optimization,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3213–3221.
- [69] S. Sridhar, F. Mueller, M. Zollhoefer, D. Casas, A. Oulasvirta, and C. Theobalt, “Real-time joint tracking of a hand manipulating an object from RGB-D input,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 294–310.

- [70] S. Melax, L. Keselman, and S. Orsten, “Dynamics based 3d skeletal hand tracking,” in *Proc. Graphics Interface*, 2013, pp. 63–70.
- [71] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon, “Learning an efficient model of hand shape variation from depth images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 2540–2548.
- [72] A. Tkach, M. Pauly, and A. Tagliasacchi, “Sphere-meshes for real-time hand modeling and tracking,” *ACM Trans. Graph.*, vol. 35, no. 6, pp. 222:1–222:11, 2016.
- [73] J. Romero, D. Tzionas, and M. J. Black, “Embodied hands: Modeling and capturing hands and bodies together,” *ACM Trans. Graph.*, vol. 36, no. 6, pp. 245:1–245:17, 2017.
- [74] D. Tang, T. H. Yu, and T. K. Kim, “Real-time articulated hand pose estimation using semi-supervised transductive regression forests,” in *Proc. Int. Conf. Comput. Vis.*, 2013, pp. 3224–3231.
- [75] D. Tang, J. Taylor, P. Kohli, C. Keskin, T.-K. Kim, and J. Shotton, “Opening the black box: Hierarchical sampling optimization for estimating human hand pose,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 3325–3333.
- [76] C. Choi, A. Sinha, J. Hee Choi, S. Jang, and K. Ramani, “A collaborative filtering approach to real-time hand pose estimation,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 2336–2344.
- [77] G. Shakhnarovich, P. Viola, and T. Darrell, “Fast pose estimation with parameter-sensitive hashing,” in *Proc. Int. Conf. Comput. Vis.*, 2003, pp. 750–758.
- [78] J. Romero, H. Kjellstrm, and D. Kragic, “Monocular real-time 3D articulated hand pose estimation,” in *Proc. IEEE-RAS Conf. Humanoid Robots*, 2009, pp. 87–92.

- [79] —, “Hands in action: real-time 3D reconstruction of hands in interaction with objects,” in *Proc. IEEE Conf. Robotics and Automation*, 2010, pp. 458–463.
- [80] C. Keskin, F. Kra, Y. Kara, and L. Akarun, “Hand pose estimation and hand shape classification using multi-layered randomized decision forests,” in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 852–863.
- [81] H. Liang, J. Yuan, and D. Thalmann, “Parsing the hand in depth images,” *IEEE Trans. Multimedia*, vol. 16, no. 5, pp. 1241–1253, Aug. 2014.
- [82] P. Li, H. Ling, X. Li, and C. Liao, “3D hand pose estimation using randomized decision forest with segmentation index points,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 819–827.
- [83] C. Wan, A. Yao, and L. Van Gool, “Direction matters: hand pose estimation from local surface normals,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 554–569.
- [84] F. Kirac, Y. E. Kara, and L. Akarun, “Hierarchically constrained 3D hand pose estimation using regression forests from single frame depth data,” *Pattern Recognition Letters*, vol. 50, pp. 91–100, Dec. 2014.
- [85] P. Dollr, P. Welinder, and P. Perona, “Cascaded pose regression,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 1078–1085.
- [86] P. Krejov, A. Gilbert, and R. Bowden, “Combining discriminative and model based approaches for hand pose estimation,” in *Proc. IEEE Conf. Automatic Face and Gesture Recog.*, 2015, pp. 1–7.
- [87] —, “Guided optimisation through classification and regression for hand pose estimation,” *Computer Vision and Image Understanding*, vol. 155, pp. 124–138, 2017.
- [88] Q. Ye, S. Yuan, and T.-K. Kim, “Spatial attention deep net with partial PSO for hierarchical hybrid hand pose estimation,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 346–361.

- [89] C. Wan, T. Probst, L. Van Gool, and A. Yao, “Crossing nets: Dual generative models with a shared latent space for hand pose estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 680–689.
- [90] H. Guo, G. Wang, X. Chen, C. Zhang, F. Qiao, and H. Yang, “Region ensemble network: Improving convolutional network for hand pose estimation,” *Proc. Int. Conf. on Image Process.*, 2017.
- [91] C. Choi, S. Kim, and K. Ramani, “Learning hand articulations by hallucinating heat distribution,” in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 3104–3113.
- [92] M. Oberweger and V. Lepetit, “Deepprior++: Improving fast and accurate 3d hand pose estimation,” in *Proc. Int. Conf. Comput. Vis. Workshop*, 2017, pp. 585–594.
- [93] X. Chen, G. Wang, H. Guo, and C. Zhang, “Pose guided structured region ensemble network for cascaded hand pose estimation,” *CoRR*, vol. abs/1708.03416, 2017. [Online]. Available: <https://arxiv.org/abs/1708.03416>
- [94] M. Oberweger, P. Wohlhart, and V. Lepetit, “Hands deep in deep learning for hand pose estimation,” in *Proc. Comput. Vis. Winter Workshop*, 2015, pp. 21–30.
- [95] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei, “Model-based deep hand pose estimation,” in *Proc. Int. Joint Conf. Artificial Intell.*, 2016, pp. 2421–2427.
- [96] A. Sinha, C. Choi, and K. Ramani, “Deephand: Robust hand pose estimation by completing a matrix with deep features,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 4150–4158.
- [97] C. Wan, T. Probst, L. Van Gool, and A. Yao, “Dense 3d regression for hand pose estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 5147–5156.

- [98] G. Moon, J. Y. Chang, and K. M. Lee, “V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 5079–5088.
- [99] M. Rad, M. Oberweger, and V. Lepetit, “Feature mapping for learning fast and accurate 3d pose inference from synthetic images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 4663–4672.
- [100] S. Baek, K. I. Kim, and T.-K. Kim, “Augmented skeleton space transfer for depth-based hand pose estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 8330–8339.
- [101] G. Poier, D. Schinagl, and H. Bischof, “Learning pose specific representations by predicting different views,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 60–69.
- [102] A. Toshev and C. Szegedy, “Deeppose: Human pose estimation via deep neural networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 1653–1660.
- [103] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, “Efficient object localization using convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 648–656.
- [104] T. Pfister, J. Charles, and A. Zisserman, “Flowing convnets for human pose estimation in videos,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 1913–1921.
- [105] J. M. V. Verth and L. M. Bishop, *Essential Mathematics for Games and Interactive Applications, Second Edition: A Programmer’s Guide*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.
- [106] W. Li, Z. Zhang, and Z. Liu, “Action recognition based on a bag of 3D points,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshop Human Communicative Behavior Analysis*, 2010, pp. 9–14.

- [107] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, “What is the best multi-stage architecture for object recognition?” in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 2146–2153.
- [108] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778.
- [109] E. Shelhamer, J. Long, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2017.
- [110] J. Xu, J. Yuan, and Y. Wu, “Multimodal partial estimates fusion,” in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 2177–2184.
- [111] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon, “The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 103–110.
- [112] R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A matlab-like environment for machine learning,” in *Proc. Neural Information Processing Systems Workshop, BigLearn*, 2011, pp. 1–6.
- [113] S. Pellegrini, K. Schindler, and D. Nardi, “A generalization of the ICP algorithm for articulated bodies,” in *Proc. British Mach. Vis. Conf.*, 2008.
- [114] O. Koller, H. Ney, and R. Bowden, “Deep hand: How to train a CNN on 1 million hand images when your data is continuous and weakly labelled,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 3793–3802.
- [115] S. Song and J. Xiao, “Deep sliding shapes for amodal 3D object detection in RGB-D images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 808–816.
- [116] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and F.-F. Li, “Towards viewpoint invariant 3D human pose estimation,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 160–177.

- [117] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 2261–2269.
- [118] Intel. (2018) Intel realsense sdk. [Online]. Available: <https://software.intel.com/en-us/intel-realsense-sdk>
- [119] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, “Kinectfusion: Real-time dense surface mapping and tracking,” in *Proc. IEEE Symp. Mixed and Augmented Reality*, 2011, pp. 127–136.
- [120] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
- [121] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Proc. Int. Conf. Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.
- [122] A. Dai, C. R. Qi, and M. Nießner, “Shape completion using 3d-encoder-predictor cnns and shape synthesis,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 5868–5877.
- [123] R. Girshick, “Fast R-CNN,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
- [124] D. Kinga and J. B. Adam, “A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: <https://arxiv.org/abs/1412.6980>
- [125] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” *CoRR*, vol. abs/1412.6806, 2014. [Online]. Available: <https://arxiv.org/abs/1412.6806>
- [126] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

- [127] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.
- [128] H. Guo, G. Wang, X. Chen, and C. Zhang, “Towards good practices for deep 3d hand pose estimation,” *CoRR*, vol. abs/1707.07248, 2017. [Online]. Available: <https://arxiv.org/abs/1707.07248>
- [129] C. Xu, L. N. Govindarajan, Y. Zhang, and L. Cheng, “Lie-X: Depth image based articulated object pose estimation, tracking, and action recognition on lie groups,” *International Journal of Computer Vision*, pp. 454–478, 2016.
- [130] Oculus. (2018) Oculus. [Online]. Available: <https://www.oculus.com/>
- [131] G. Riegler, A. O. Ulusoy, and A. Geiger, “Octnet: Learning deep 3d representations at high resolutions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 3577–3586.
- [132] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 652–660.
- [133] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in *Proc. Conf. Neural Information Processing Systems*, 2017.
- [134] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, “Spatial transformer networks,” in *Proc. Conf. Neural Information Processing Systems*, 2015.
- [135] H. Fang, S. Xie, Y.-W. Tai, and C. Lu, “RMPE: Regional multi-person pose estimation,” in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 2334–2343.
- [136] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learning Representations*, 2015, pp. 1–15.

- [137] J. H. Friedman, J. L. Bentley, and R. A. Finkel, “An algorithm for finding best matches in logarithmic expected time,” *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226, 1977.
- [138] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, “Surface reconstruction from unorganized points,” *Computer graphics*, vol. 26, no. 2, pp. 71–78, 1992.
- [139] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, “Joint training of a convolutional network and a graphical model for human pose estimation,” in *Proc. Neural Information Processing Systems*, 2014, pp. 1799–1807.
- [140] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, “Convolutional pose machines,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 4724–4732.
- [141] A. Newell, K. Yang, and J. Deng, “Stacked hourglass networks for human pose estimation,” in *Proc. European Conf. Comput. Vis.*, 2016, pp. 483–499.
- [142] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, “Coarse-to-fine volumetric prediction for single-image 3d human pose,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 7025–7034.
- [143] G. Rogez, P. Weinzaepfel, and C. Schmid, “LCR-net: Localization-classification-regression for human pose,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 3433–3441.
- [144] D. Tome, C. Russell, and L. Agapito, “Lifting from the deep: Convolutional 3d pose estimation from a single image,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 2500–2509.
- [145] G. Rogez, J. S. Supancic, and D. Ramanan, “Understanding everyday hands in action from rgb-d images,” in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 3889–3897.
- [146] T.-H. Pham, N. Kyriazis, A. A. Argyros, and A. Kheddar, “Hand-object contact force estimation from markerless visual tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.