Learning to Estimate 3D Human Pose from Point Cloud

by

Yufan Zhou

Thesis submitted to the University of Ottawa
In partial Fulfillment of the requirements for the
M.A.Sc. degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science, Faculty of Engineering,
University of Ottawa
Ottawa, Canada

Abstract

As the development of the health and well-being industry advances, the importance of maintaining physical exercise on a regular basis should be understated. To help people evaluate their pose during exercise, pose estimation has aroused huge interest among researchers from vary fields. Meanwhile, pose estimation, especially 3D pose estimation, is a challenging problem in computer vision. Although a lot of progress has been made over the past few years, there are still some limitations, such as low accuracy and the lack of comprehensive and challenging datasets for use and comparison. In this thesis, we study the task of 3D human pose estimation from depth images. Different from the existing CNN-based human pose estimation method, we propose a deep human pose network for 3D pose estimation by taking the point cloud data as input data to model the surface of complex human structures. We first cast the 3D human pose estimation from 2.5D depth images to 3D point clouds and directly predict the 3D joint position. Our proposed methodology combining a two-stage training strategy is crucial for pose estimation tasks. The experiments on two public datasets show that our approach achieves higher accuracy than previous state-of-art methods. It reaches the accuracy of 85.11% and 78.46% on both part of the ITOP dataset, and the accuracy of 80.86% on the EVAL dataset.

Acknowledgements

I would like to thank my supervisor, Prof. Abdulmotaleb El Saddik, for his patient guidance, encouragement, and advice during my time as his student. At many stages of my research, I benefited so much from his advice and explored lots of new ideas. I have been extremely lucky to have a supervisor who cared so much about my studies and research. His excellent guidance and positive outlook in my research inspired me and gave me confidence.

Likewise, I would also thank the members of MCRLab for their help and thoughtful comments. My sincere thanks go to Haopeng Wang for his kindly supports and suggestions. In particular, I would like to thank Dr. Haiwei dong for his insightful suggestions and guidance throughout my research.

Finally, I would like to thank my friends for their encouragement. I am grateful to my family for supporting me spiritually throughout writing this thesis, study, and life in general.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation of the problem

The vision of a digital twin as stated in [43] by Prof. El Saddik is a digital replica of a living or non-living physical entity. By bridging the physical and the virtual world, data is transmitted seamlessly allowing the virtual entity to exist simultaneously with the physical entity. A digital twin continuously learns and updates itself from the external environmental condition, using multi-sensor to monitor, understand, and optimize the functions of the physical entity and providing continuous feedback to improve quality of life and well-being. Therefore, a digital twin is the convergence of several technologies

such as Internet of Things, Artificial Intelligence, Machine Learning, Cyber Security, and Communication Networks.

Pose estimation is a specific part of the AI technology for digital twin, which is to identify and locate the key joints of the human body (head, left hand, right foot, left hip, etc) in an image. Because digital twin analyzes, recognizes and imitates human behavior through the estimated key joints, the key joints of the human skeleton are extremely important to describe human pose and predict human motions for digital twin. Therefore, the detection and location of human skeleton key joints have played a fundamental role in the computer vision system of digital twin. Besides, the high-level applications of pose estimation are mainly focused on patient monitoring system, human-computer interaction, virtual reality, human animation, smart home, intelligent security, athlete-assisted training and so on. For example, as a specific application for human-computer interaction by bridging the physical and the virtual worlds, Microsoft Kinect has achieved an excellent performance in the Xbox 360 gaming system. Kinect uses a depth camera to capture the player's body movements and recognizes the positions of different parts of the human body in the three-dimensional space.

In the field of human pose estimation, the directions in the past decade can be summarized in three different points. First, based on different inputs, it can be divided into two approaches: depth and color images. Besides, pose estimation can be divided into multi-person pose estimation and single-person pose estimation. The difficulty of multi-person pose estimation is greater than that of single. In addition, based on the different tasks, it can be divided into two directions: 2D and 3D. 2D pose estimation is to detect the key points of human body in the color image and display them in the coordinate system (u, v) of a 2D color image, while the 3D pose estimation is to detect the key points of human body on the depth image and display them in the coordinate system (x, y, z) of a 3D depth image.

3D pose estimation based on depth images has been widely used in related fields of 3D computer vision, such as motion capture and augmented reality. Most applications

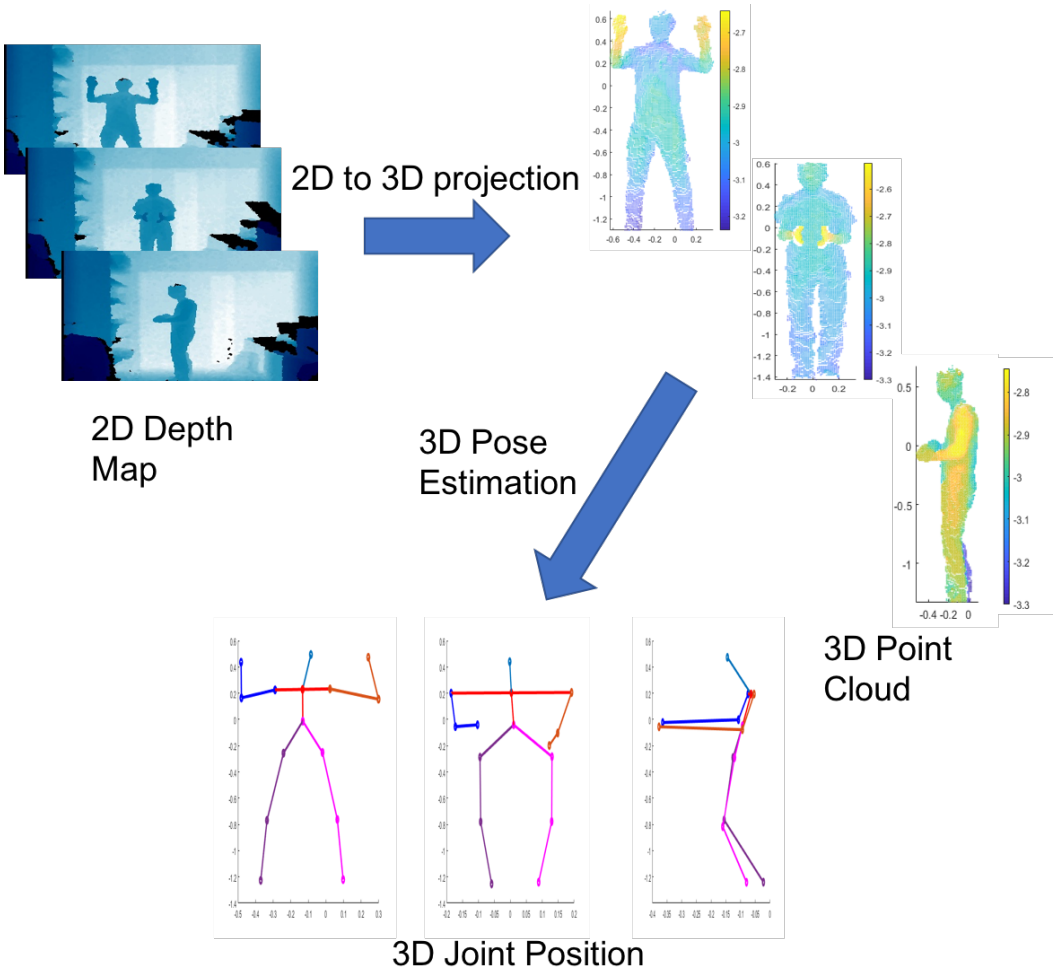

Figure 1.1: 3D point cloud has a one-to-one relation with a 3D pose. Our approach is based on point clouds, converting depth images into point clouds before pose estimation.

for depth-based pose estimation were based on machine learning, such as random forest. Kohli *et al.*[37] proposed a real-time tracking system by using a Kinect and random forest algorithm to estimate the human pose in an indoor environment. Kinect was considered as a perceptual part of their human-computer interaction systems to capture depth maps. The motions were reconstructed in the virtual environment by joint positions so that the system presented a better understanding of human movement. Others took depth maps as input images to process 3D estimation by using deep learning model like [6], [14], [13]. In 3D computer graphics, the image channel of a depth map contains information about the distance between the surface of the viewpoint and the object. It is similar to a gray-scale image, except that each pixel value of it is the actual distance of the sensor from the object. Using depth maps for pose estimation is a new trend in the future. However, current methods treated depth maps as 2D images. Researchers used convolution network to direct process the images and extracted the specific features form the depth maps.

As the depth map represents a 3D information, converting the depth maps into point clouds is another trend for process the 3D information data. The process of 3D pose estimation based on converting 2D depth images into 3D point clouds is shown in Figure 1.1. Moon *et al.*[33] took 3D voxelized grids converted from point cloud as input and estimated the per-voxel likelihood for each key point. A point cloud refers to a set of points in a three-dimensional space. Point clouds are generated by 3D scanners, such as lidar(2D/3D), stereo camera, time-of-flight camera. These devices measure the information of a large number of points on the external surfaces of objects around them in an automated way. With the widespread use of radar and depth cameras in robotics and autonomous driving, the study of point cloud has gradually been improved from the geometric feature extraction to high-level understanding. Recent works of PointNet [38], PointNet++[39], and Dynamic Graph CNN [54] performed both 3D object classification and 3D object segmentation on point clouds directly.

1.2 Challenges for 3D human pose estimation based on point cloud

The general challenge of human pose estimation is that various postures and shapes will appear since the human body is quite flexible. A new pose will be produced after there is a small change in any part of the human body. At the same time, the visibility of its key points is greatly affected by clothing, posture, viewing angle, the effects of occlusion, light, fog, and other environments. Also, there exists the visual foreshortening effects in different parts of the body. Therefore, estimating the key points of the human body or skeleton has become a very challenging task in the field of computer vision.

In addition, there are some challenges in processing point clouds. Point cloud is a set of vectors in a three-dimensional coordinate system. These vectors are usually represented in the form of X, Y, Z three-dimensional coordinates and generally used primarily to represent the outer surface and shape of the object. In addition to the geometric position information represented by (X, Y, Z), the point cloud can also represent the RGB color, gray value, depth, segmentation, etc. The challenges of the pose estimation from the point cloud are concluded as follows:

- The data structure of the point cloud is a set of points consisting of point coordinates in three-dimensional space. It is essentially a low-resolution resampling of the three-dimensional geometry, so it can only provide one-sided geometric information.

- The sparsity of the point cloud has restricted the power of the neural network. When the same object is scanned by different devices in different positions, the order of the three-dimensional points varies widely.

- It is difficult to output the 3D position of human body key joints directly in high accuracy since the current approaches regress the heatmaps to achieve an excellent performance by using deep learning models.

Although there are lots of the datasets for human pose estimation, most of them only include 2D color images with 2D or 3D joint positions, such as MPII [1], COCO keypoint dataset [30], FLIC [3], FLIC Plus[49]. For 3D human motion datasets, there are also so many available datasets with depth maps for 3D pose estimation, such as ITOP [14], Human3.6M [22], EVAL [10] and CAD-60/120 [47], [27]. We use two datasets for our approach: EVAL and ITOP, which are discussed in Chapter 4. There are also some problems with the 3D human pose dataset. First, some datasets are lack of enough depth images with its corresponding human key body joints. Second, there is no uniform standards or rules to label the skeleton joints. Different datasets are annotated with different labels. Commonly used skeleton joints for labeling are hands, head, chest, spine, torso, hips, knees, feet, elbows, shoulders and neck. It's hard to fuse these data from different datasets for training and testing in deep learning model. Second, 3D skeletons joints of most datasets are captured from the Kinect system instead of using the Vicon camera system to track the markers with high accuracy.

1.3 Objectives

The objectives of our thesis are summarized as follows:

- Propose a deep learning model to estimate the human body pose from depth images. A new approach based on point cloud has been provided for directly regress the 3D joint positions.

- Propose a new training method to optimize the networks that combines a supervised two-stage training strategy. The employment of this method achieves accuracy superior to that of current state-of-the-art methods on two available public datasets.

1.4 Thesis Statement

Estimating the 3D joints from images using the CNN-based model have achieved a high accuracy from 2015 to now. In our work, we aim to propose a network to directly regress the 3D body joint position from point clouds. We take the point cloud converted from depth maps as input to our model. Inspired by the outstanding performance of the convolutional neural network (CNN) in feature extraction, we use a dynamic graph CNN to process the point clouds. For normalization of points from point cloud and joints regression, a transform network part and a regression part are proposed for our model. Besides, to solve the irregularities of point clouds, a fine-tuning strategy is used to train our spatial transform network. The proposed method was evaluated on the two public datasets: ITOP [14] and EVAL [10] and achieved competitive results, showing in Chapter 5.

1.5 Contribution

The contributions of our thesis are summarized as follows:

- Design and development of a model to estimate the human body key joints directly from 3D point clouds. Unlike other methods that regress the 3D key points from the depth images, we cast the problem of 3D pose estimation from a single depth image to the point clouds.
- Development of a two-stage training strategy in our approach is proposed to optimize a spatial network for eliminating point cloud irregularities and improve the performance of our network.
- Comprehensive evaluation configurations for two existing representative 3D human pose datasets are carried out to provide a baseline for valid comparisons with other CNN-based [14] and RF/RTW-based [44], [24], [6] human pose estimation

methods from depth images. Experimental results show that our network for 3D pose estimation has a significantly accurate performance.

1.6 Thesis Outline

The remainder of the thesis is organized as follows:

- Chapter 2 elaborates on the background and related work of human pose estimation and point cloud processing from conventional methods to novel state-of-the-art deep learning related methods.
- Chapter 3 presents the methodology used in our proposed approach and the detailed training process of our pose estimation network.
- Chapter 4 introduces the details of the datasets used in this work and evaluation metrics.
- Chapter 5 provides different experiments on two public datasets, showing the excellent results and the effectiveness of our proposed approach. Also, this chapter discusses the issues appearing during this work.
- Chapter 6 summarizes the merits of our proposed approach, presents the limitations, and discusses planned future work.

Chapter 2

Background and Related Work

The main purpose of our work is to estimate key joints of a human body, and the following parts focus on human pose estimation from a single image. This is a multi-faceted task that includes target detection, pose estimation, segmentation, and more. Pose estimation can make the way for computers or devices to have a better understanding on human behaviors. Therefore, it will improve the future human-computer interaction system. Human pose estimation includes both two-dimensional (2D) and three-dimensional (3D) human pose estimation. Application in the 2D pose estimation have already achieved excellent results using various methods from previous researches. In contrast, applications in 3D human pose estimation still have a great potential of improvement in terms

of process time and accuracy.

After the brief introduction on the human pose estimation task in Section 2.1, we introduce the related work in color-based pose estimation and depth-based pose estimation methods in Section 2.2 and 2.3. Next, we focus on the background of 3D deep learning for point cloud classification and segmentation and present the edge convolution operation for point cloud with a related dynamic graph CNN model in Section 2.4. Finally, Section 2.5 reviews the transfer learning method for network optimization.

2.1 Human Pose Estimation Task

Human pose estimation plays a fundamental role in the research of other related fields of computer vision, such as behavior recognition, character tracking, gait recognition, and many other related fields. An example of human pose estimation based on a color image has shown in Figure 2.1, where the position of each joint is accurately estimated in the example. It aims to identify and locate the key points of all human bodies on the image. This can be a basic research topic for many visual applications such as human motion recognition and human-computer interaction. Specifically, practical applications with human pose estimation can be develop into intelligent video surveillance, patient monitoring systems, virtual reality, human animation, smart home, smart security, and athlete-assisted training.

In computer vision, the task of pose estimation is to reconstruct the joints and limbs of the person based on the image, which can propose various challenging due to uncontrollable conditions. The visibility of the key joints is greatly affected by wearing, posture, and viewing angle. The main challenges include detecting people and locating their key points simultaneously without giving a person's location when testing, and the difficulty lies in reducing the complexity of the model analysis algorithm in adapting to various changes.

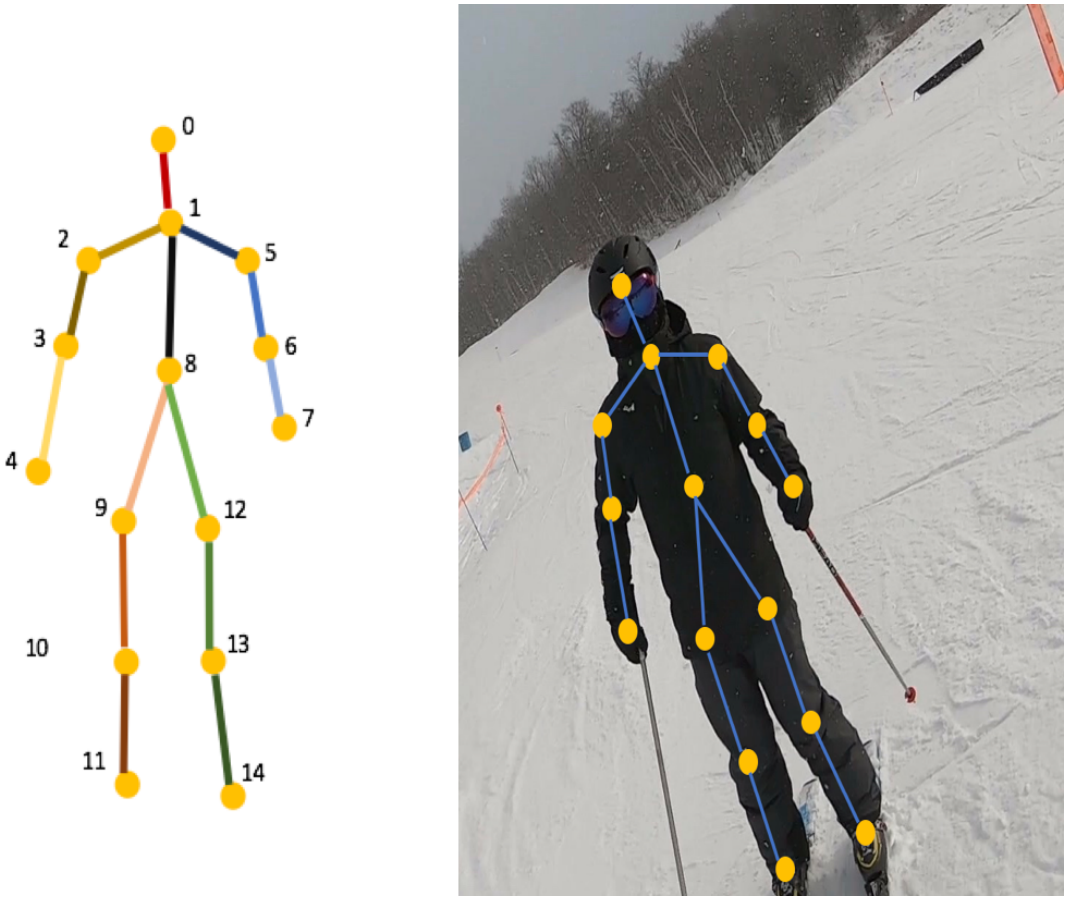

Figure 2.1: An example of the pose estimation. The left image lists 15 key joints, while the right image is the result from estimating the human pose on a color image. The joints include head, neck, torso, shoulders, elbows, hands, hips, knees and feet.

2.2 color-based pose estimation method

Human body pose estimation based on the RGB images has broad application prospects in the fields of behavior recognition, human-computer interaction, games, animation, etc. In general, the estimated results using color images as inputs represent the position of the human skeleton in a two-dimensional image coordinate system. With the significant development of the neural network in recent years, the estimation of the body's key points has been continuously improved. Toshev *et al.* [51] directly regress joint coordinates, with a cascade of ConvNet regressors to improve accuracy over a single pose regressor network. Due to the complexity and flexibility of human movement, it's a low accuracy result for direct regression of key point positions. Pfister *et al.* [36] regarded pose estimation as a detection problem and used optical flow information to regress a heatmap. Yang *et al.* [60] [59] designed a Pyramid Residual Module (PRMs) to enhance the performance of their model. As a branch of human pose estimation, 3D humane pose estimation has been gradually developed into a higher level. According to a different input, there are existing color-based 3D pose estimation and depth-based 3D pose estimation. Zhou *et al.* [65] and Ke1 *et al.* [25] used a two-stage cascaded structure to estimate 3D pose in the wild. Wang *et al.* [53] presented the 2D-to-3D pose regression in two-stage methods.

Typically, the approaches for 2D multi-person pose estimation can be divided into two types: the top-down approach and the bottom-up approach. In the top-down approach, a person detector is followed by a pose estimator on each person. The other approach is to detect all body parts from a multi-person image and then regroup these body parts afterwards. In this section, we introduce the most significant color-based 2D pose estimation methods and then compare their difference. AlphaPose [63] is one of the traditional top-down methods for pose estimation, which is an effective solution since their main idea is to create a combination of Single Shot MultiBox Detector (SSD) [31] and Stacked Hourglass estimator [35]. Besides, Single Shot Multi-box Detector with convolutional pose machines [55] represents the most classical pose estimation method

among the top-down structures because the pose estimation is performed on a region where the person is located. Therefore, this type of method is dependent on the accuracy of the person detector. Errors in locations of the person will lead to a lowered accuracy on estimation. Additionally, Mask RCNN [16] is a popular architecture for performing detection, segmentation, and multi-person pose estimation. Detection is performed on each generated candidate region. When it detects the region contains a person, the position of each key point on the human body is then uniquely encoded. The processing time is directly proportional to the number of people processed by theses top-down methods. Also, these approaches have a large disadvantage – the accuracy of estimated results depends heavily on the performance of the person detector. Furthermore, if the detector can not find any person from the image, it is impossible for the estimator to output any joint position.

DeepPose [46] is one of the earliest CNN-based models for the human pose estimation. A multi-staged architecture is proposed for the direct regression of the joint coordinates. This basic concept of multi-staged architecture has inspired more recent 2D pose estimation architectures, such as Stacked Hourglass [35]. Their approach shows that Stacked Hourglass can achieve ideal results by constantly refining heatmaps due to several reasons. Firstly, the Stacked Hourglass architecture has made effective use of the residual connections. Also, the final predictions (heatmaps), which predict the occurrence of specific joints at each pixel-level, are generated by the successful use of pooling and upsampling. The achieved high accuracy is demonstrated by experimental results on FLIC [3] and MPII human pose dataset [2]. Thus, their solutions to improve human pose estimation using the deep learning model to generate heatmaps have proven to produce sound performances. The concept of heatmap on pose estimation was first proposed by [50]. The ground-truth heatmap is generated by using a two-dimensional Gaussian function in the ground-truth coordinate system in order to output a two-dimensional predicted heatmap. As shown in Figure 2.2, the CNN produces a heatmap as a two-dimensional image. During the training, the loss is calculated by using the mean squared error be-

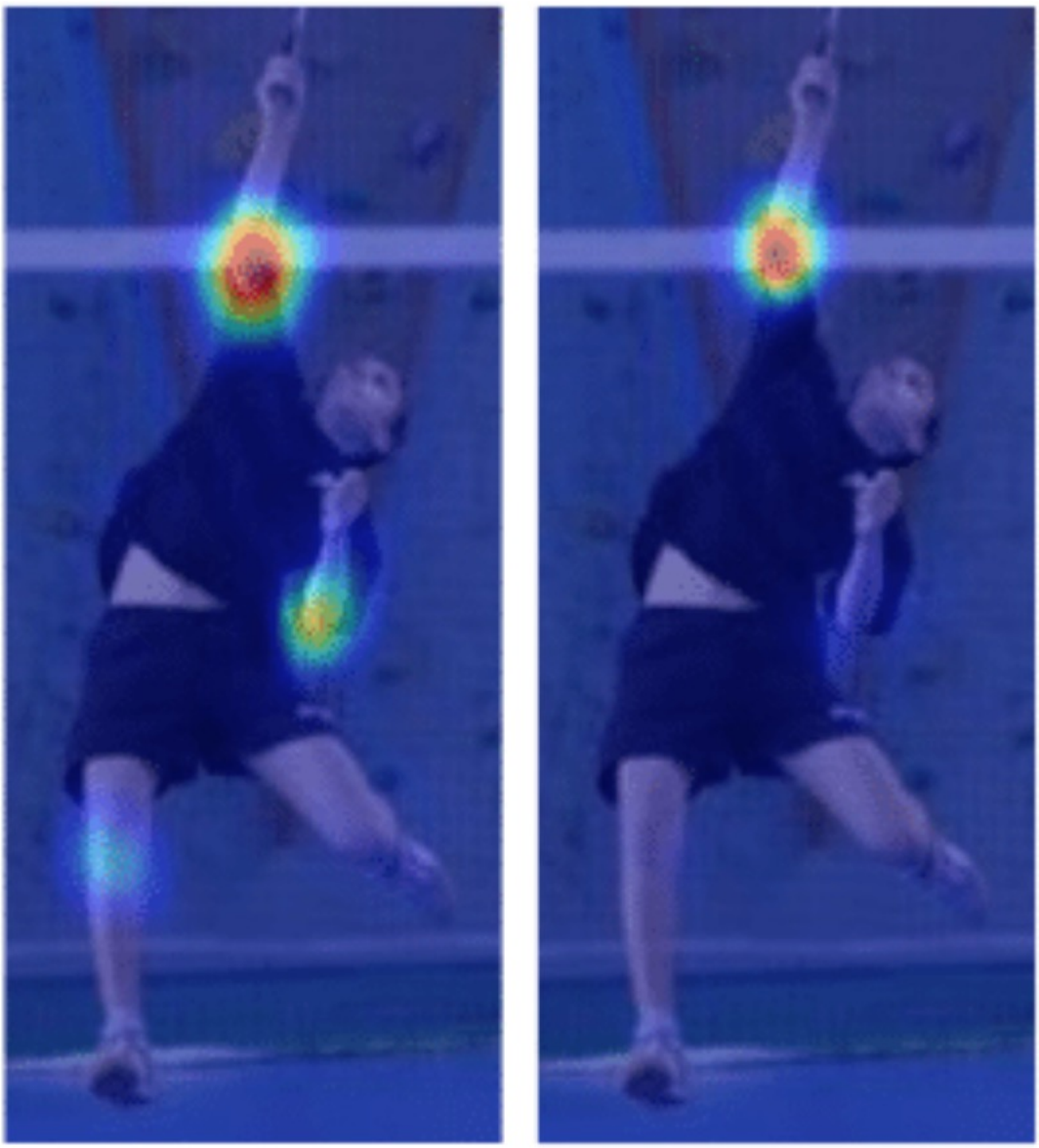

Figure 2.2: An example of heatmap on the position of elbows, which is from the last stage of CNN from convolutional pose machine [55].

tween the output heatmap and an ideal target: a 2D spherical Gaussian mean-centred on the ground-truth joint location. Finally, the 2D coordinates of the outputs are obtained through a non-maximum suppression method [41]. In Equation 2.1, the ground-truth is changed into a form of the heatmap.

$$f(x, y) = \exp - \frac{(x - c_x)^2 + (y - y_c)^2}{2} \tag{2.1}$$

Where x and y are the image coordinates; c_x and c_y are the ground-truth joints in the image coordinate.

Inspired by generating heatmaps for joint prediction, Xiao et al. [56] created an modified network with added deconvolutional layers over the last convolution stage based on the ResNet architecture. More specifically, their network generated heatmaps from low-resolution images with the help from three additional deconvolutional layers , batch normalization, and ReLU activation [32]. Another idea for the task of pose estimation is to convert pose estimation into a sequence problem, such as pose machine [40]. As a sequence predictive framework, pose machine can learn spatial information to capture the interactions between body parts. Convolutional pose machines (CPMs) [55] integrated convolutional network into pose machines to learn image features and image-dependent spatial models to estimate human pose.

Real-time pose estimation is the key component in the robotic field for machines to understand human motion. Cao et al. [5] proposed a bottom-up human pose estimation network named OpenPose using Part Affinity Fields (PAFs). PAFs is a 2D vector set, and each 2D vector encodes the position and the orientation of a limb. These fields and joint confidence maps are together learned and predicted by CNN. OpenPose can effectively detect 2D poses of multiple people from an image in the real time. Different from the traditional top-down structure, OpenPose adopted the method of detecting the joint and then performing the bottom-up of matching. It is not sensitive to the number growth of people. The time cost is mainly concentrated on the complexity of the CNN network model. The usages of PAFs have greatly improved the speed of the multi-person pose estimation.

2.3 depth-based pose estimation

Compared to color images, depth maps contain 3D information about the distance between scene object and the camera viewpoint. Therefore, the results of 3D human body pose estimation using single color images are not ideal. In this section, we present the current methods based on 3D pose estimation using depth maps.

The methods of depth-based 3D pose estimation are divided into two parts: generative and discriminative models. The generative models are similar to the template matching. The human body templates are required to find correspondence between the inputs and templates in generative models. On the other hand, discriminative models directly estimate the key joints of human body parts. The details of the two models are discussed in the following part.

The traditional human key joints detection is based on the principle of geometric-prior template matching. The core idea of template matching is to use the template to represent the whole human body structure, including the representation of key points, the representation of limb structure, and the representation of the relationship between different limb structures. A ideal template matching can simulate large range of poses so that it can more accurately match and detect the corresponding human posture. Iterative closest point algorithms [10], [12], [18], [19] are commonly used for 3D body tracking, which are motivated by the point clouds from depth sensors. Similar to the RGB based methods, to further constrain the output space, graphical models [17] imposed kinematic constraints to improve full-body pose estimation.

Different from generative models, discriminative models directly estimate the pose of the body. Most of discriminative models are based on random forests (RF). Shotton et al. [44] classified body parts from a single depth image based on the random forest classifier. Jung et al. [24] used a random tree walk algorithm (RTW) to regress the joint position. Besides, there are various CNN-based methods for the 3D pose estimation. Haque et al. [14] proposed a viewpoint-invariant model using CNN and recurrent networks for

the human pose estimation, while Guo et al. [13] introduced a tree-structured region ensemble network for the 3D position regression. These deep learning models are based on the image features as they can directly take depth images as the input. Although there are numerous of advantages for CNN to process images, to effectively extract a full 3D information from the depth maps effectively is nearly impossible since the depth maps are still 2.5D. Therefore, converting them to 3D data is a new direction. A point-cloud-based method which takes the point cloud as input named voxel-to-voxel network (V2V-PostNet) was proposed in [33]. For each voxel, the network estimated the likelihood of each body joint. V2V-PostNet then extracted the 3D joint positions from the generated heatmaps. In addition, in the field of 3D hand pose estimation, Ge et al. [11] creatively proposed Hand PointNet, taking the point cloud from the depth image of hand parts directly and processing the set of points to output the 3D joint positions. The network accurately regressed a low dimensional representation of the 3D hand pose. Simultaneously, they also proposed a refinement network to further improve the accuracy of the estimation.

Cai et al. [4] predicted a 2D joint position from the RGB images and estimated the 3D pose after calculating depth image patches with the corresponding prediction of the 2D joint locations. Their operation is to simply combine the depth maps and color images, which can still be categorized as a 2D pose estimation. The key point is how to directly use the depth image by extracting the required features from images and fusing them into a neural network for training. Moreover, Zimmermann et al. [50] used a depth image to assist in 3D human pose estimation and fused the voxel of depth maps into the proposed network.

2.4 point-cloud-based method

In the past few years, there has been a lot of in-depth research on two-dimensional and achieved considerable development, especially for the classification tasks. Similarly to the

successful achievement of 2D image processing, the development of 3D data processing has become more and more exciting with the help of 3D interaction devices such as Intel RealSense, Microsoft Kinect sensor, and other types of depth/IR cameras. These devices have a positive impact on the technology of human-computer interaction (HCI) [7], [58]. Images captured by the depth sensors are widely used in robotics and autonomous driving. Meanwhile, the depth data can be converted into point clouds. The flexible geometric representation provided by point clouds has a wide application in computer graphics. For instance, point cloud segmentation and classification using a deep learning model is a new and challenging field for computer vision. Also, the study of point cloud maps has gradually improved from the geometric feature extraction to high-level understanding such as point cloud segmentation [34] and recognition [38]. In this section, we introduce 3D deep learning from the point cloud.

A series of PointNet models, including PointNet [38] and PointNet++ [39] are the recently proposed methods for point cloud classification and segmentation. Point clouds are fundamentally irregular, and it is not sensitive to the order of the data. This means that the model for processing point cloud data needs to be invariant to different permutations of the data. A spatial transform network named T-Net (part of PointNet [38]) has been designed to ensure the invariance of the model to a specific spatial transformation. The points are first aligned by multiplying it with a transformation matrix learned by a spatial transform network. PointNet models treat each point individually, learning the mapping from 3D to potential features without taking advantage of geometry. A single maxpooling layer is used for all the features of sampled points in PointNet. This operation loses many local features and only maintains global features. Therefore, the network's ability to extract local information from the model is far less satisfactory than that of the convolutional neural network. PointNet ++ extracts features on different scales and obtains deep features through a multilayer cascade network. There are three main modules, sampling, grouping and feature learning, for extracting both local and global features. However, PointNet++ still considers individual points in local point sets

instead of the relationships between a pair of points.

In contrast to previous networks, Wang et al. [54] successfully developed a dynamic graph CNN model (DGCNN) to process point clouds based on a thinking way of image analysis using a convolution neural network (CNN). The method is the same as the principle of a convolutional network that considers the set of interconnected pixels instead of a single pixel. Edge Convolution, as the main part of DGCNN, takes a center point with its nearest neighbor points as an input to the multilayer perceptron (MLP) layer. The Edge Convolution layer models the distance and geometric structure between points when constructing the neighborhood. Local features are formed by searching neighbor points, while global features can be extracted through stacking or recycling Edge Convolution layers. However, unlike PointNet and PointNet++, not only global features but also local features are considered in Edge Convolution layers.

2.4.1 DGCNN Model

Point cloud segmentation and classification by using deep learning model to process 3D point clouds remain a challenging problem in computer vision. A straightforward way to apply deep learning on a point cloud is to transform the data into a volume representation. For example, a 3D voxel grid has been widely used in the field of 3D deep learning. We can train a CNN with a 3D filter directly without any neural network problem in the use of voxel grid data. However, the main disadvantage of volume data is a large amount of computation. If there is a typical image with a size of $128 \times 128 = 16384$ pixels, there is $128 \times 128 \times 128 = 12097152$ volume data after converting the 2D dimension to a 3D dimension which is a large amount of data causing very slow processing time. Therefore, the required solution is a direct deep learning approach that will focus on 3D coordinates instead of volume data.

Instead of using volume data, the DGCNN model can directly use the point cloud as input. At the same time, the model gets enough local information for classification, segmentation, and other tasks. There are two parts in DGCNN model–classification and

segmentation. The classification network consists of two Edge Convolution layers, followed by a pooling operation and three full connection layers, and then the classification results are obtained. The segmentation network uses three Edge Convolution layers, followed by three full connection layers. In the DGCNN model, shape information and features can be extracted by stacking the Edge Convolution module or by using it cyclically. From the calculation process of Edge Convolution, it can be seen that when KNN is used to find k nearest points for each feature update, the result of each KNN step will be different because it is based on the distance of the new feature, and the local graph will be updated dynamically every iteration.

DGCNN exploited local structures by constructing a local neighborhood graph and design the convolution-like operations on the edges connecting neighboring pairs of points. The excellent contribution of the DGCNN is the first one to propose the geometric relationships among the points from the point cloud. Although learning from 3D coordinates directly is a difficult problem, DGCNN extracts features from geometric information formed by coordinates. In the original Euclidean space, DGCNN considers the local features from K nearest neighbors by modeling the distance between neighborhood points. The local features represent the geometric structure between neighborhood points and the final feature called edge feature in neural network computing is a fusion of global and local features.

2.4.2 Edge Convolution

Lecun et al.[29] has successfully established a modern structure of CNN, which is widely used in many computer vision tasks, such as image processing, image classification, object tracking, and so on. The core of CNN is called 'kernel' or 'filter' working as a feature detector. By sliding these filters over images, these detectors produce the feature maps which contain relative location information, object shape information, etc. The convolution layer is also known as the local feature learning layer, where the input images are connected with this local area of the receptive field. The basic process of

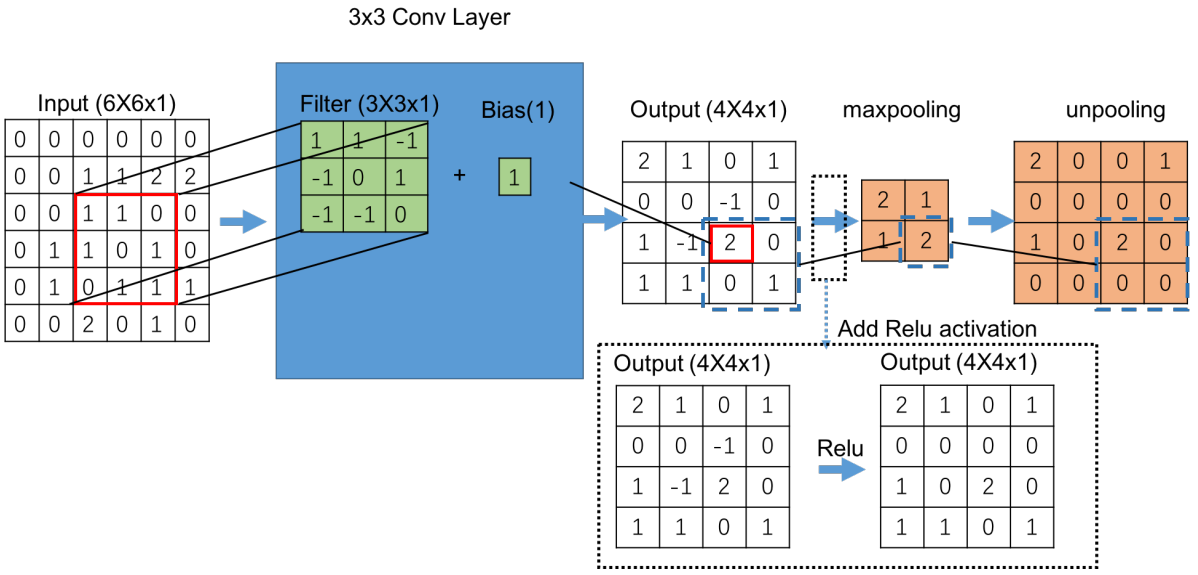

Figure 2.3: The basic process of CNN filter with Relu, Maxpooling, and Unpooling. The blue box is the convolution layer; 3x3 is the kernel size. Relu is an activation function which is to add nonlinear factors. Maxpooling is to choose the maximum value from each 2x2 block. Unpooling is a common up-sampling method.

CNN is shown in Figure 2.3. The filter chooses the 3×3 matrix from the input volume through a certain stride (which we defined as 1 in Figure 2.3). The value 1 of the output in the red box is passed by each value in the matrix and multiplies the corresponding weights in the filter. The output of convolution layer is processed by pooling layer which is used to reduce the size of feature map and the parameters in the network. But for point cloud, the discontinuity of point cloud distribution in 3D makes the neighborhood structure of point cloud difficult to deal with by using CNN.

Inspired by the convolution neural network (CNN) [29], Edge Convolution layer (EdgeConv) is proposed as a basic part of DGCNN [54] to solve the neighborhood feature extraction directly from point cloud. It is a new neural-network module suitable for CNN-based high-level tasks on point clouds including classification and segmentation. The appealing property is that the EdgeConv incorporates local neighborhood information as it can be stacked or recurrently applied to learn global shape properties. The extracted features from EdgeConv layers are named edge features which are the relation-

ship between neighborhood points. The previous model DGCNN showed that the results have been improved by using these edge features

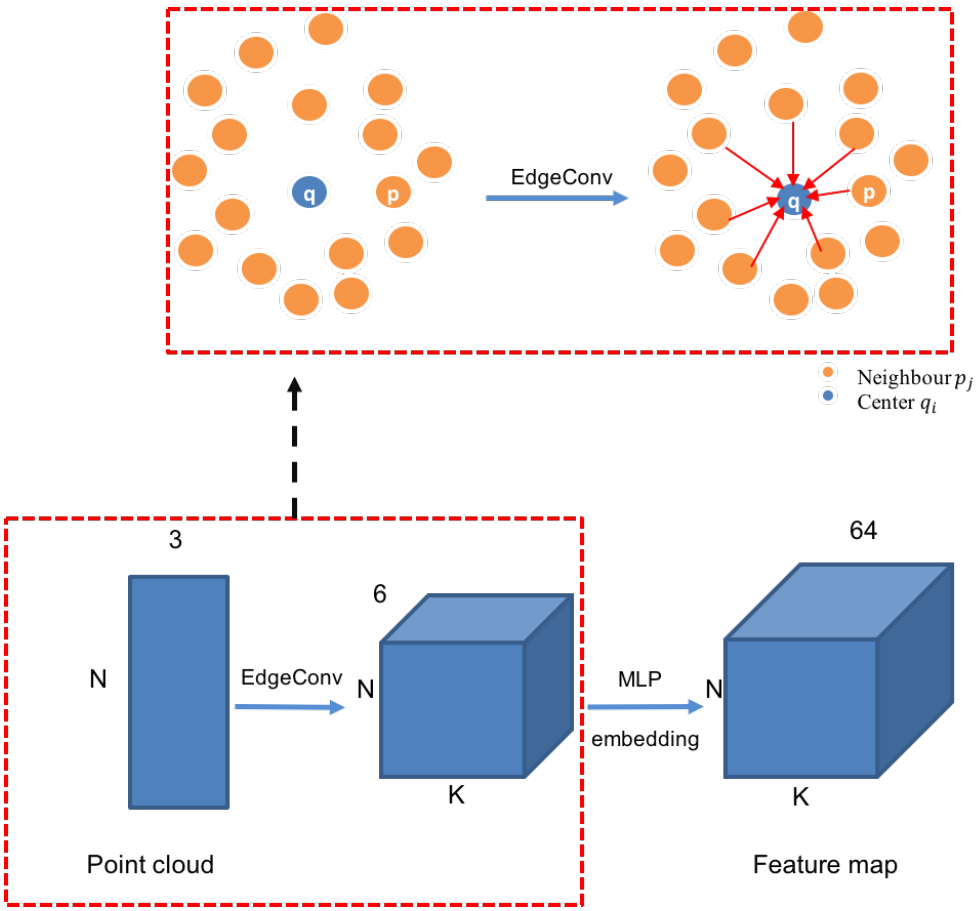

Figure 2.4: An example of edge features in the operation of EdgeConv. There is a set of 17 points. In the middle is the center point. Find the nearest 5 neighborhood points based on the center point in this figure. The red line represents the edge feature between the two points.Edge feature can be represented as a vector $\vec{pq}$. q is the center point, while p is one of its 5 neighborhood points. After embedded by a multi-layer perceptron, there are 64 generated feature maps.

EdgeConv considers the neighbor points as the independent local features and captures local geometric structure while maintaining permutation invariance. Edge features are generated to the relationship between a point and its neighbors instead of generating points' features directly from embedding. The EdgeConv is designed to better capture

local geometric functions and be invariant to the ordering of neighbors and thus permutation invariant. Also, EdgeConv applies channel-wise symmetric aggregation operation [62] on the edge features associated with all the edges emanating from each vertex. In 3D point cloud computing, point-to-point relationships are always within a global scope of thinking. The operation of the point-to-point relationships by using EdgeConv are shown in Figure 2.4. The size of input is $N \times 3$, while the neighborhood features are $N \times K \times 64$ after the operation of EdgeConv. Embedding process is a convolution kernel of 1×1 which is used to enhance the dimension of the features, and $N \times K \times 64$ is obtained. Besides, the geometric structure between a center point q and its neighborhood point p is represented by a vector $\vec{pq}$. EdgeConv models the distance between two points (which is the norm of $\vec{pq}$). However, when constructing the neighborhood, and it ignores the direction of the vector, which leads to loss of local geometric information.

2.5 Transfer Learning

Transfer learning [61] is a machine learning method that refers to a pre-trained model being reused in another task. The pre-trained model means that the model has been trained on a large benchmark dataset. By using the pre-trained model on a big data set, we can directly use the corresponding structure and weight to apply them to the problems we are facing. As a popular method in deep learning, it allows us to build accurate models in a time-saving way. Through the meaning of transfer learning, the knowledge acquired in one environment is used in another to improve its generalization performance. For example, in the case of processing a large number of resources required to train a deep model or having a large number of data sets for pre-trained models. Transfer learning performs well only when the features from the pre-trained model in the first task is a generalization feature. In general, as a method of optimization, we can save time for training and have a better performance with the help of transfer learning. However, using transfer learning in a new domain does not benefit until the model is

developed and evaluated.

In computer vision, it is common to apply transfer learning in predictive modeling problems that take images or videos as input. It is common practice to use pre-trained models for large-scale challenging image datasets in case of predictive modeling problems. For example, the Imagenet dataset [9] has been widely used as a training set because it is large enough (including 1.2 million images) to help train the model of universal value. Some classic network structures are used for pre-training on Imagenet, such as VGG [45], GoogleNet[48], MobileNet [20], AlexNet [28], ShuffleNet [64], DenseNet [21] and ResNet [15]. Many researchers apply the weights of these pre-trained models to their models and deepen the training again to achieve the best results on their datasets. When the process of transfer learning, these pre-trained models also show good generalization performance for images outside Imagenet dataset. This method is effective because a large number of pictures are trained on this model, and the model needs to be able to predict a relatively large number of image types. Conversely, this requires the model to effectively learn to extract features from the image to better solve the problem.

Since the pre-trained model has been trained in an excellent situation, there is no need for us to modify too many weights in a short time. When we use it in transfer learning, it is often just fine-tuned. Transfer learning is a machine learning technique while applying deep learning is fine-tuning. Selectively load pre-trained network model weights by modifying the structure of the pre-trained network model structure (such as modifying the number of sample category outputs). In general, the first few layers are trained to identify features of the task. More often in practice, fine-tuning the existing networks by training on the smaller datasets achieves the effect of a fast training model. Assuming that our dataset is not very different from the context of the original dataset, the pre-trained model will learn the features associated with our classification problem. The way to use the pre-trained model in various situations is shown in Figure 2.5. There are two main factors to judge: the size of the new dataset and its similarity to the original dataset. The learning rate is normally set low when we fine-tune our

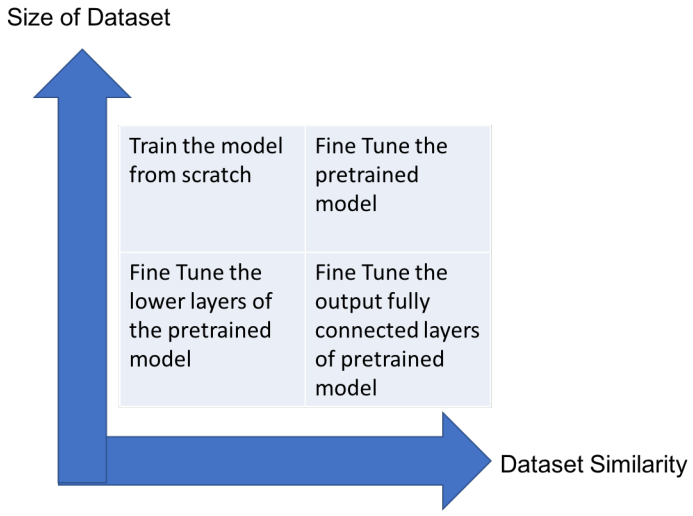

Figure 2.5: The method train the model is determined by the size of the dataset and the similarity between the datasets.

network. Training methods in four situations: First, the small size of the dataset and high similarity compared with the training data of the pre-trained model. We can use the pre-trained model as the feature extractor and transform the output layer into a structure that fits the problem situation. Second, the small size of the dataset and low data similarity. We can freeze the weights in the first k layers of the pre-trained model, and then retrain the next $n-k$ layers, where n is the total number of the layer from the network. Next, the big size of the dataset and low data similarity. Because there is a big difference between the actual data and the training data of the pre-trained model, using the pre-trained model will not be an efficient way. Last, the large size of the dataset a high data similarity. This is the ideal situation, using the pre-trained model will become very efficient. The best way is to keep the original structure and initial weight of the model unchanged, and then retrain based on the new data set. In summary, as a concept of transfer learning, fine-tuning is an effective way of adjusting our network to get good results with new problems. The main methods for fine-tuning are taking the network as feature extraction, training the specific layers after freezing the other layers, and using a pre-trained model.

2.6 Summary

As introduced before, there are lots of 2D/3D pose estimation approaches based on the deep learning model, but most of them can not directly regress joint positions. EdgeConv is usually applied to process point clouds. Inspired by this, a point cloud containing pose information can be fed into a deep learning model based on EdgeConv to estimate the key joints, which is a regression task. Also, a spatial transform network from PointNet is used to maintain the permutation (order) invariance of the point cloud. By combining EdgeConv and spatial transform network, a more appropriate module is produced to extract features from the point cloud and directly estimate the 3D key joint position. In the next chapter, the details of our human pose net and preprocessing for point cloud will be introduced independently.

Chapter 3

Methodology

Our method for pose estimation takes the point clouds converted from the depth images as input, considering this F-dimensional point cloud with N points. In this section, the point cloud is defined as a set of vectors $P = \{p_1, p_2, ..., p_N\}$. Outputs from our model are a set of 3D key joint positions $J = \{j_1, ..., j_M\}$, where M is the number of key body joints and j_i contains x_i, y_i, z_i in the camera coordinate system where $i \in \{1, ..., M\}$. The input to the deep learning model is (P, J). Considering the resolution of the depth image, we set N as $5,000$ and F as 3. To further improve the results, our network is built on a modified DGCNN model, which takes these sampled point clouds to regress the 3D body pose by extracting the global and local features.

In this chapter, we introduce the methodology of our proposed 3D pose estimation model. Our 3D pose estimation model is named human pose net. As we reviewed in chapter 2, lots of related work performed well in 3D pose estimation. They directly conducted training on the images and regressed the heatmap, or simply used the random forest to accomplish this task. However, most of them didn't consider the depth maps as the real 3D information. As shown in Figure 3.1, we covert the depth maps into the point clouds. The human pose network takes the point cloud as input, while this point cloud is preprocessed by removing a large number of discrete points. After presenting the method for point cloud preprocessing, we propose our human pose network based on a modified DGCNN. Our 3D pose estimation model contains two parts–spatial transform network, two units and three fully connected layers. In addition, a two-stage training strategy is proposed to optimize our spatial transform network.

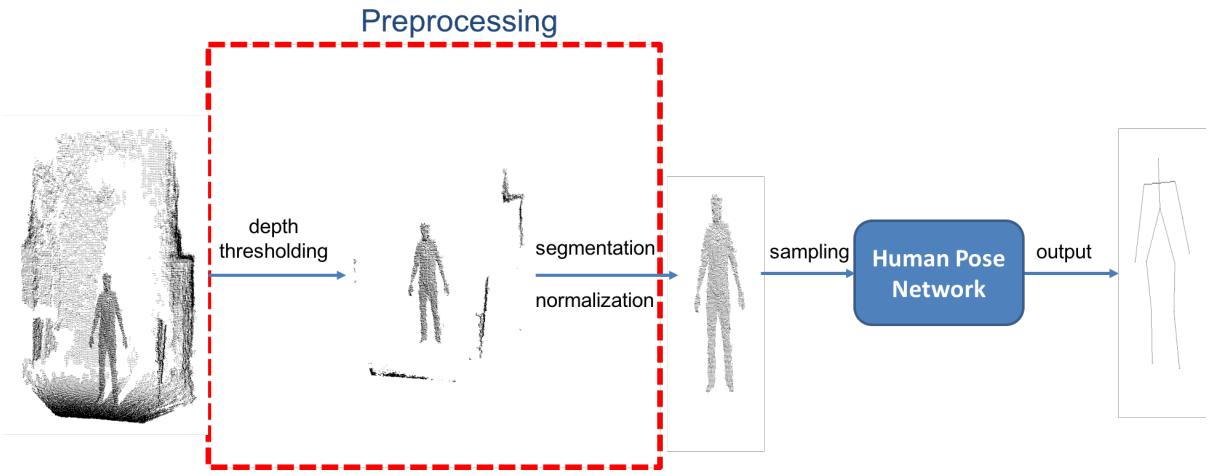

Figure 3.1: Left is the point cloud which is converted from the depth image. We define distance thresholds to cut the subject out and segment the human body from the background. After sampling the point cloud into the same size, input it to the regression network. The output is the 3D coordinates of the skeleton joints.

3.1 Preprocessing

In this section, we will introduce our method for preprocessing: segmentation and normalization.

3.1.1 Segmentation

First, our approach segments the human body from the background as clearly as possible. Figure 3.1 shows the process of segmentation and sampling. Since the data are captured by the depth camera, many points with invalid depth values or zero-depth values may be included. Therefore, we remove all invalid and zero points. Given the depth information, most of the background points can be easily removed by defining depth thresholding. However, only setting the depth thresholding is not enough to extract the human body. Point clouds still include noise and other background objects, which may affect the results. These are mostly due to the photon shot noise and long distance between the human body and the camera.

Setting a bounding box and depth thresholding can eliminate many background objects from Fgiure 3.1. In a more general case, we can make use of nearest neighbors and implement a clustering technique that is essentially similar to a flood fill algorithm [8]. The main idea for removing background subjects in point clouds is using the Euclidean cluster extraction filter. Since point cloud data provide higher dimensional data, there is considerable information that can be extracted. In our proposed approach, segmentation based on Euclidean distance was performed by removing the small noise cluster. Euclidean cluster extraction algorithm uses distance between neighbors as a criterion. The human body always remains the biggest cluster. As shown in Algorithm 1, it is compared to the distance between the points. Each point from the point cloud is used to check whether the distance between it and its neighbors is below a thresholding. If the distance is less than the threshold (radius), point and its neighbors are treated as the same cluster. K-nearest neighbors are selected from the samples to be clustered before a

cluster can no longer spread. Next, we continue in the set of remaining points and repeat the above steps to find a new cluster until the end of the traversal. The thresholding is set to 0.1 meters. Here is the Euclidean cluster algorithm for extracting the human body from the backgrounds.

Algorithm 1 Euclidean Cluster Extraction

Data: point cloud P

Result: the point cloud of human body with background removed C_{body}

Set up an empty list of clusters C, a cluster of human body C_{body}, a queue of the points that need to be checked CH, and a distance thresholding d_{th}.

for *every point* $p_i \in P$ **do**

add p_i to the current queue CH

for *every point* $p_i \in CH$ **do**

search for the neighbors of p_i, and take the neighbors as a set P_k^i. The search range is a sphere with radius $r < d_{th}$

check if every neighbor point $p_j^i \in P_k^i$ has already been processed. If not, add it into CH

end

When all points in CH have been already been processed, add the cluster queue CH into the list C and set the CH to empty.

end

$C_{body} = Argmax_{x \in C} h(x)$

3.1.2 Normalization

After the previous steps, considering the different numbers of points, we downsampled the number of original point clouds into an average size (5000 points) since the number of points from preprocessed point cloud is between 4000 and 7000. Normalization of the point cloud is executed by the person's height (b_h) and width (b_w) in this point cloud to address the data of different sizes. Since joint coordinates are in absolute image coordinates, it proves beneficial to normalize them with a bounding box b, which includes

the person's height (b_h) and width (b_w). In a trivial case, the box can denote the full image. Such a box is defined by its center. The center in the point cloud is the human body center b_c. b_c can be derived from the average of all joints from the ground truth. As shown in Equations 3.1 and 3.2, $NOR(p_i, b)$ is the function for point cloud normalization.

$$NOR(p_i, b) = (p_i - b_c) \begin{bmatrix} \frac{1}{b_w} & 0 & 0 \\ 0 & \frac{1}{b_h} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$b_c = \frac{\sum_{i=0}^{N} p_i}{N} \tag{3.2}$$

where N is the number of points from the human body and p_i is the point from point cloud P of the human body.

3.2 Human Pose Network

In this section, we present our proposed deep learning model for pose estimation at first. As a basic part of DGCNN, the EdgeConv layer mentioned in Chapter 2 is widely used in our proposed network. After our pose estimation network, we show our optimization function and two-stage training strategy for our deep learning model.

We design a human pose network that can directly estimate the 3D coordinates of the pose from the input point clouds by modifying DGCNN and PointNet. In our model, we use T-Net as the spatial transform network to achieves permutation invariance of points and stack EdgeConv layers to extract and learn features from the transformed point clouds. The original DGCNN model is used for classification and segmentation. We modified the architecture of the classification model by changing the middle layers and fully connected layers. The architecture of the modified model is shown in Figure 3.2. Different from the original model, we use one EdgeConv layer with 128 filters instead of three 64 filters to keep a suitable receptive field and reduce the parameters due to the limitations of our computer performance. EdgeConv is a basic part of the DGCNN,

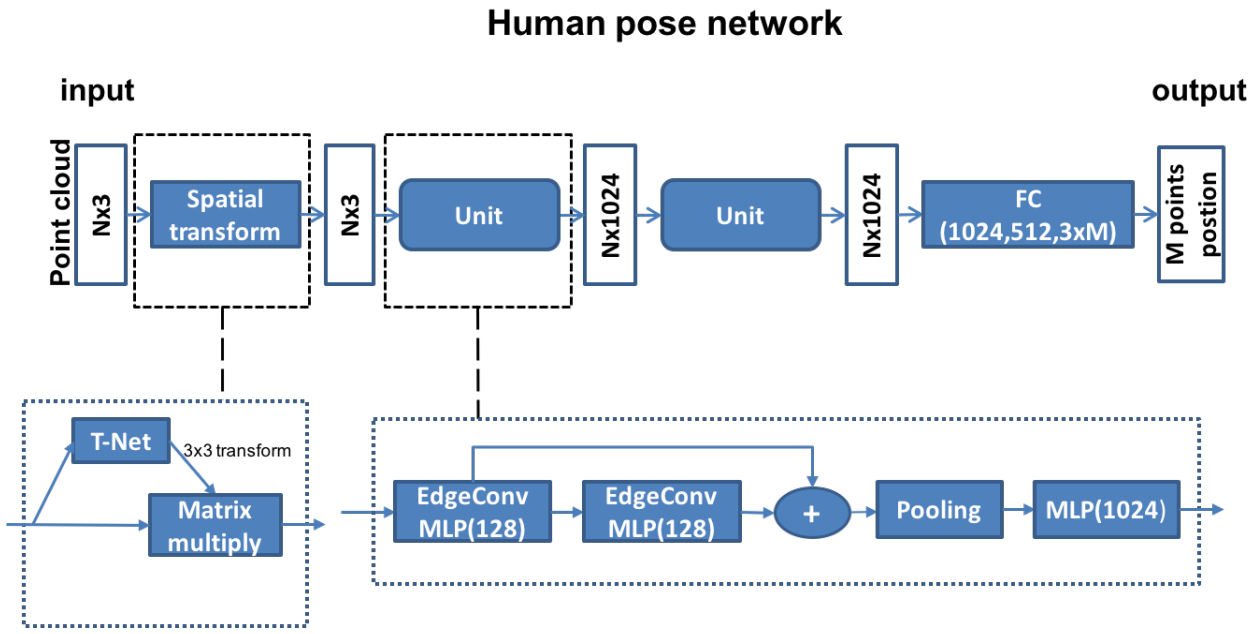

Figure 3.2: The architecture of human pose network. The normalized 3D point clouds are fed into the regression network. The size of input is the $N \times 3$ point clouds while the size of output is $M \times 3$. M represents the number of key point positions. The normalized 3D point cloud are input to $N \times 3$. Our network is trained in an end-to-end manner to extract hand features and regress 3D joint locations.

which captures the local structures of the human body by combining the points and their neighborhoods. In our model, EdgeConv has been used as a main part for feature extraction. As Figure 3.3 shows, we visualize an edge feature that is considered a local feature. The feature is composed of the K nearest neighbors by calculating the Euclidean distance. EdgeConv takes the local feature as input, considering the coordinates of points and the distance from the domain points as local domain information. Global shape information can be extracted by stacking or recycling EdgeConv layers.

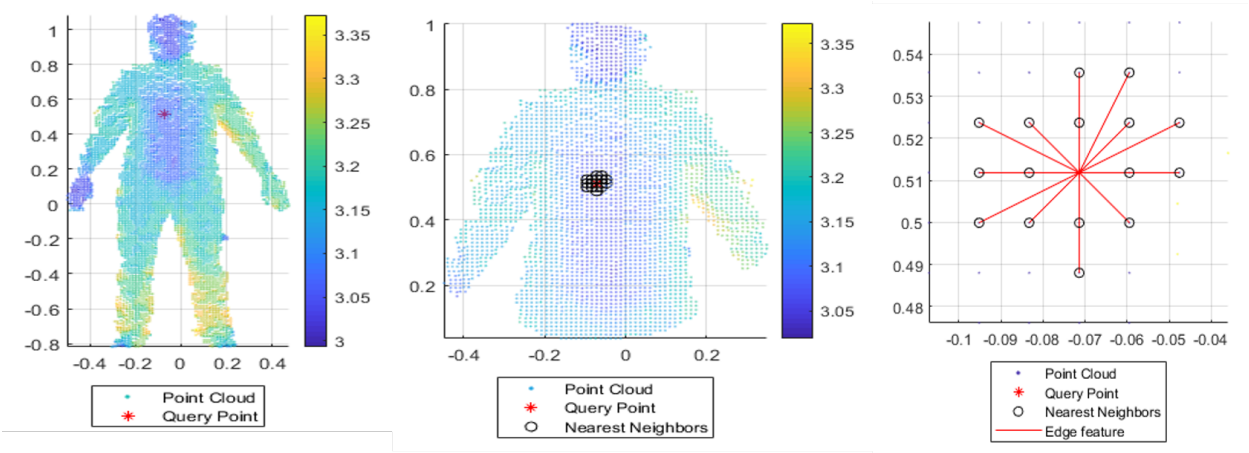

Figure 3.3: An example of edge features from the chest. We define an arbitrary center query point from the chest and its neighborhoods. Depending on the distance, we can find the K neighbor points. The edge features are composed of these $K+1$ points. The point clouds with edge features are input to the neural network. In this figure, we set K to 16.

For each point $p_i \in P$, we find its K neighbor points and concatenate the neighbor points with the original point cloud as the edge features, which are calculated in the EdgeConv layers. There are two units in our network, which are composed of EdgeConv layers. The first unit is composed of two EdgeConv (128 filters) layers that are connected with MLP layers (1024 filters). Between the EdgeConv layer and an MLP layer is a pooling layer. In our network, we use a global max-pooling layer to reduce the dimension of data and aggregate point features. The input to the MLP layers is the concatenation

of the outputs from the first two EdgeConv layers. The concatenation is considered as a kind of shortcut connection, which is a classic operation of Resnet [15]. This allows data streams to flow across layers. In the process of training, the network can deepen the understanding of the details through shortcut connections. The second unit is the same as the first unit. To regress the joint positions, we modify the last three fully connected layers into 1024, 512, $3 \times M$. Since we target the single human object, the transform network is considered to be very suitable. Although the point clouds converted from depth images are ordered, points become an unordered form after preprocessing. The spatial transform network aligns an input point set to canonicalize into a specific space by applying an estimated 3×3 transformation matrix. To estimate the 3×3 matrix, the network uses the coordinates of each point in the point cloud and the coordinate difference of its K neighbors. After matrix multiplication, the pose network takes the transformed point cloud as input.

3.3 Network Training

The process of training is composed of two parts. First, after preprocessing, the point cloud is sampled into a set of $5,000$ points. We trained the whole model with the randomly arranged point cloud as the input in each step. In this way, the spatial transform network is trained in a good situation. Second, we maintain the weights of the spatial transform network and train the remaining networks. The input to the model is also a set of 5,000 points but not randomly sampling every time. The input to our network is a set of normalized points $X^{nor} = \{x_i^{nor}\}_{i=1}^{N} = \{p_i^{nor}\}_{i=1}^{N}$, where p_i^{nor} is 3D coordinates of the normalized point and the ground truth $Y^{nor} = \{y_i^{nor}\}_{i=1}^{M} = \{j_i^{nor}\}_{i=1}^{M}$, where j_i^{nor} is the corresponding key body joints after normalization. The loss function is mean square error. Given T training samples with normalized point cloud, we solve the minimization

problem:

$$\omega^* = \underset{w}{\operatorname{argmin}} \sum_{t=0}^{T} ||Y_t^{nor} - F(X_t^{nor}, \omega)||^2 + \lambda||\omega||^2 \tag{3.3}$$

where ω denotes the parameters of our network, F represents the human pose network, and λ is the regularization strength.

3.3.1 Two-stage training Scheme

The main problem for point cloud in deep learning is irregular arrangement in 3D space. We have set a spatial transformer network to ensure the model's invariance to a particular spatial transformation. The spatial transformer network is the same as the T-net from PointNet [38]. However, to achieve a higher performance of the spatial transform network, we have built a two-stage training scheme. In this section, we discuss and present our strategy of two-stage training for proposed human pose network.

The spatial transform network tries to generate a 3×3 matrix which is used to align a set of points into a canonical space. Learning through a large number of point clouds in an irregular space, spatial transform network estimates a suitable 3×3 matrix and applying the matrix on the point clouds. In the first stage of our training scheme, we expanded the domain of data through randomly shuffling the points from point cloud in each training step. These randomly sampled point clouds are arranged in different irregular orders. In the second stage, we migrate the parameters of spatial transform network from the pre-trained model to this new model. After freezing the weights from the spatial transform network, the remaining layers are trained to optimize the results of pose estimation. When training the new model, we set a low learning rate. As shown in Figure 3.4, the input to network in the second stage is different from the first stage. The point cloud data without randomly sampling is input to the our model.There is no need to sample the point cloud repeatedly.

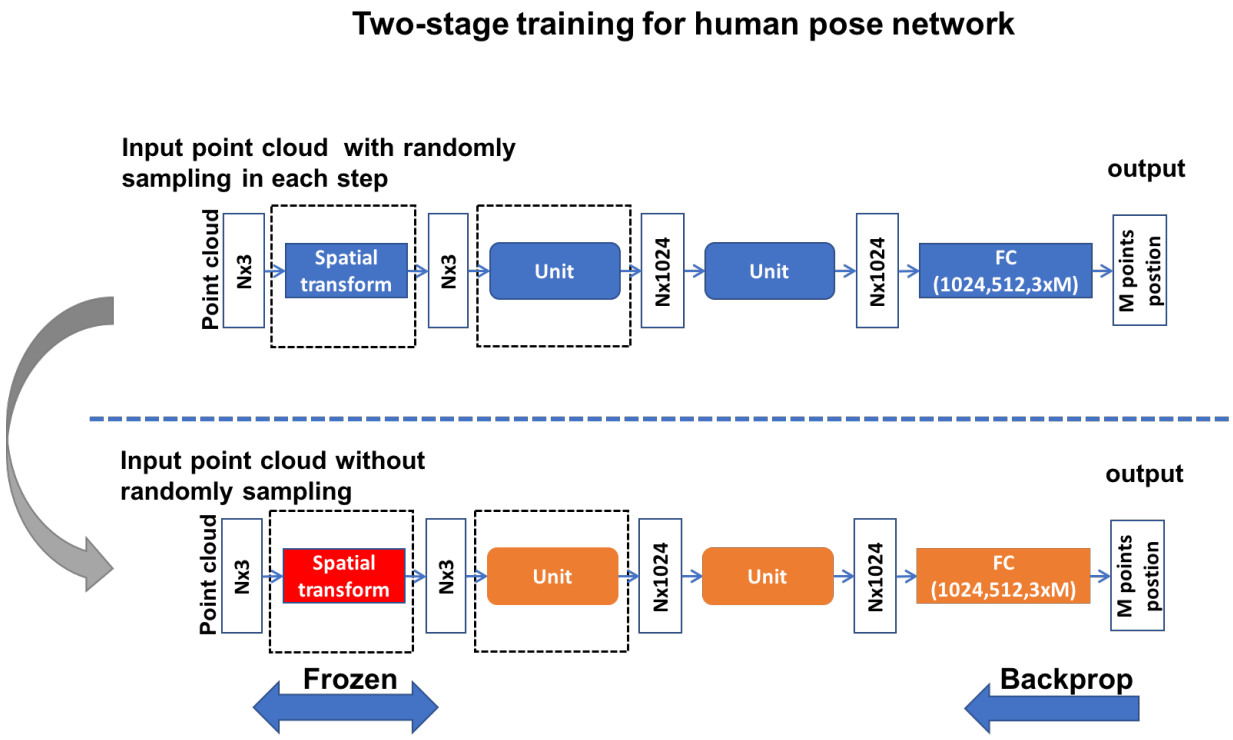

Figure 3.4: The two-stage training scheme for our proposed human pose network. The specific steps are summarized in this figure. Training the whole network is the first step. Freezing the spatial transform network and training the remaining layers are the next step. The red color means the frozen layers, while the orange color means the layers to be trained.

3.4 Summary

In this chapter, we introduced our methodology for 3D human pose estimation. Our methodology can be divided into 3 parts– preprocessing, human pose network, and our network training. In preprocessing, we removed the most background objects and segment the human body. After segmentation, a clean human body point cloud data are input to our human pose network for regressing the joint positions. To enhance the performance of our network, we proposed a new two-stage training strategy in network training.

Chapter 4

Datasets and Evaluation Mertrics

In this chapter, we introduce the evaluation metrics and two commonly used public datasets for our model training and evaluation. The evaluation metrics are presented in Section 4.1. There are several widely used datasets for human pose estimation, such as LSP [23], FLIC [3], MPII [2], COCO [30], CAD60/120 [47] [27],ITOP [14] and EVAL [10]. Among them, ITOP and EVAL are two public datasets for 3D human pose estimation. Moreover, both depth maps and 3D coordinates of key human body joints are included in these two datasets. The details of these two datasets are introduced in Section 4.2 and 4.3.

4.1 Evaluation metrics

Because pose estimation from the image is to locate the position of the joints in the human body, it is super crucial to evaluate the predicted position of the human joints with appropriate metrics. The main evaluation metrics is mean Average Precision (mAP), which is a necessary indicator to measure results in the field of computer vision for multi-label tasks. Also, OKS based on mean average precision is used in 3D pose estimation and reported for individual body parts. The evaluation metrics of the key joints from the human skeleton are analogous to the general object detection evaluation method, and the final mAP (mean average precision) value is used as an evaluation basis. OKS is object keypoint similarity which is evaluating the accuracy of human skeleton key points for researchers in the task of human skeleton key detection task. The similarity between the location and the actual annotation is scored by the distance between ground-truth and predicted joints in 3D pose estimation. A successful detection occurs when the euclidean distance in 3D space between estimated joints and ground truth is less than a threshold. The threshold is set to 10 centimeters in many related study [14], [24] and hence we will use it to compare the effectiveness of our proposed method. This rule means there is a successful detection when the predicted joints are less than $10cm$ from the ground truth. Here is the evaluation metrics.

$$AP(x,y) = \begin{cases} 1, & \text{if } distance(x,y) < 10cm \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$mAP = \frac{\sum_{i=0}^{M} AP(J_i, GT_i)}{M} \tag{4.2}$$

where $distance(x,y)$ denotes the Euclidean distance between x, y in 3D space; J and GT are predicted joints and ground truth separately. M is the total number of joints.

4.2 ITOP

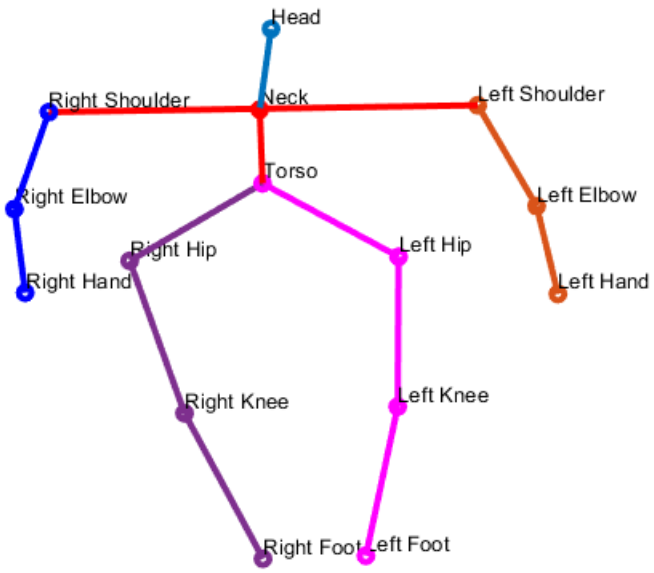

Figure 4.1: A view of joints from ITOP dataset. There are 15 key joints, which are head, neck, torso, shoulders, elbows, hands, hips, knees and feet.

The ITOP dataset [14] was established by Haque et al. in 2016. There are two view angles for the ITOP dataset when collecting the data with two Asus Xtion PRO cameras. One of the depth cameras was placed on the opposite side of the viewpoint to capture the front human motion images, while the other camera was placed at the top to capture motions with a facing-down viewpoint. Meanwhile, both front and top view angle are challenging for the task of human pose estimation based on depth maps.

As the ITOP 3D human pose estimation dataset consists of two angles of view–front-view and top-view tracks, each track contains a list of about 40k training and 4k testing depth images. The resolution of the image is 320 $\times$ 240 (width $\times$ height). The dataset also contains the point cloud which is converted from the depth maps. There are 20 actors who perform 15 sequences. As shown in Figure 4.1 and Figure 4.2, the ground-truth of this dataset is the 3D coordinates of 15 body joints, including head, neck, shoulders,

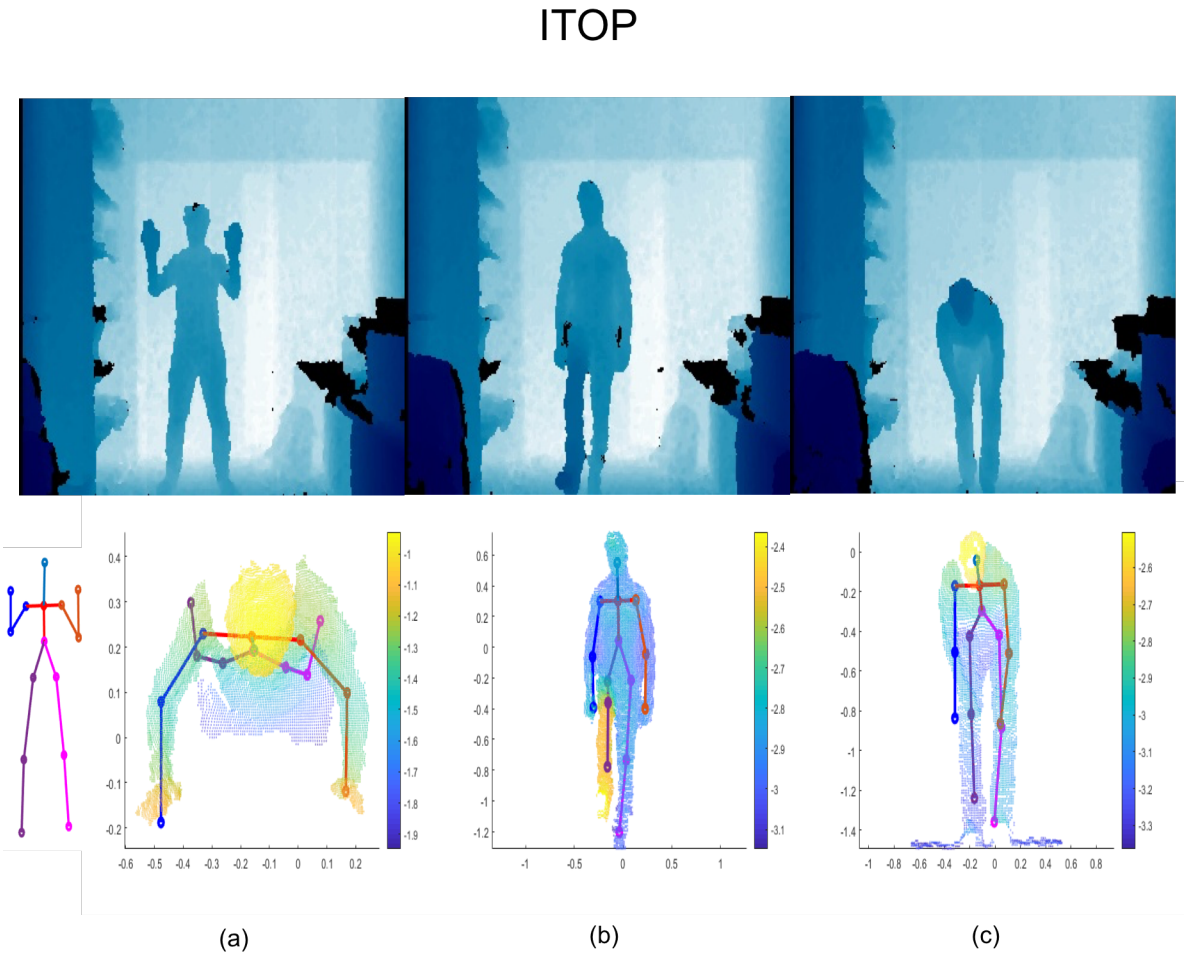

Figure 4.2: An example of ITOP dataset. The depth maps and their corresponding point cloud are shown in this figure. The color bar shows the distance between the point cloud and camera. Figure (a) is a top-view image, while the figure (b) and (c) are the front-view images. In order to show the figure (a) more clearly, the viewpoint of skeleton information is converted into the front view. The motions in the figure are hand waving, bending, kicking.

elbows, hands, torso, hips, knees, feet. The 3D joint position is directly from the camera interface. The motions in ITOP dataset contain hand waving, bending, kicking, standing up, squatting down, turning around and walking, etc.

4.3 EVAL

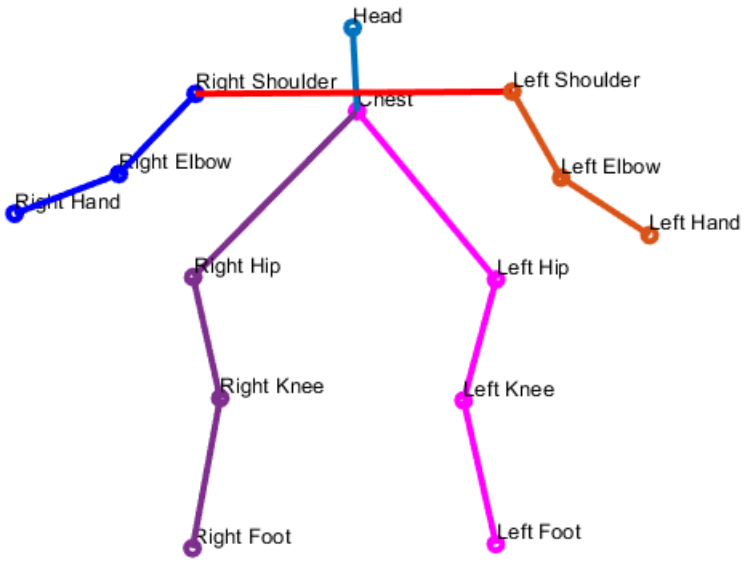

Figure 4.3: A view of joints from EVAL dataset. There are 14 key joints, which are head, chest, shoulders, elbows, hands, hips, knees and feet.

EVAL dataset [10] was established by Ganapathi et al. in 2012. The dataset contains the front-view images which are collected by the Microsoft Kinect camera at approximately 30 FPS (frames per second) and Vicon motion capture system. An example of the Vicon system is shown in Figure 4.4. The Vicon motion capture system is a set of network-connected Vicon cameras and other devices which are applied to real-time online or offline motion capture. Vicon system captures the human motion by tracking artificial markers attached to the human body. The location of human body joints is composed of these artificial markers. A stable and accurate tracking of the artificial markers is

Figure 4.4: An example of a Vicon system [52]. The model in the picture is wearing special clothing attached to the markers at the joint for system tracking.

performed with the assist of the Vicon system.

There are three subjects (one female and two males) in the EVAL 3D human pose estimation dataset. The dataset contains about 10k frames. Each frame is combined with the Vicon data of 30 markers. The Vicon data is 3D positions in the coordinate of the Vicon system. The resolution of the image is 320 $\times$ 240 (width $\times$ height). These depth maps are stored as the form of point cloud in the EVAL dataset. As shown in Figure 4.3 and Figure 4.5, the ground-truth of this dataset is the 3D coordinates of 14 body joints, including chest, hips, shoulders, elbows, knees, feet, head, hands. Each joint is set up with different Vicon markers, chest with 1 marker, hip with 2 markers, shoulder with 2 markers, elbow with 2 markers, knee with 3 markers, feet with 3 markers, head with 1 marker, hand with 3 markers. We select the average value of markers for each joint. Different from the ITOP dataset, the coordinates of joints are calculated from the Vicon system and markers. These joints are of high accuracy.

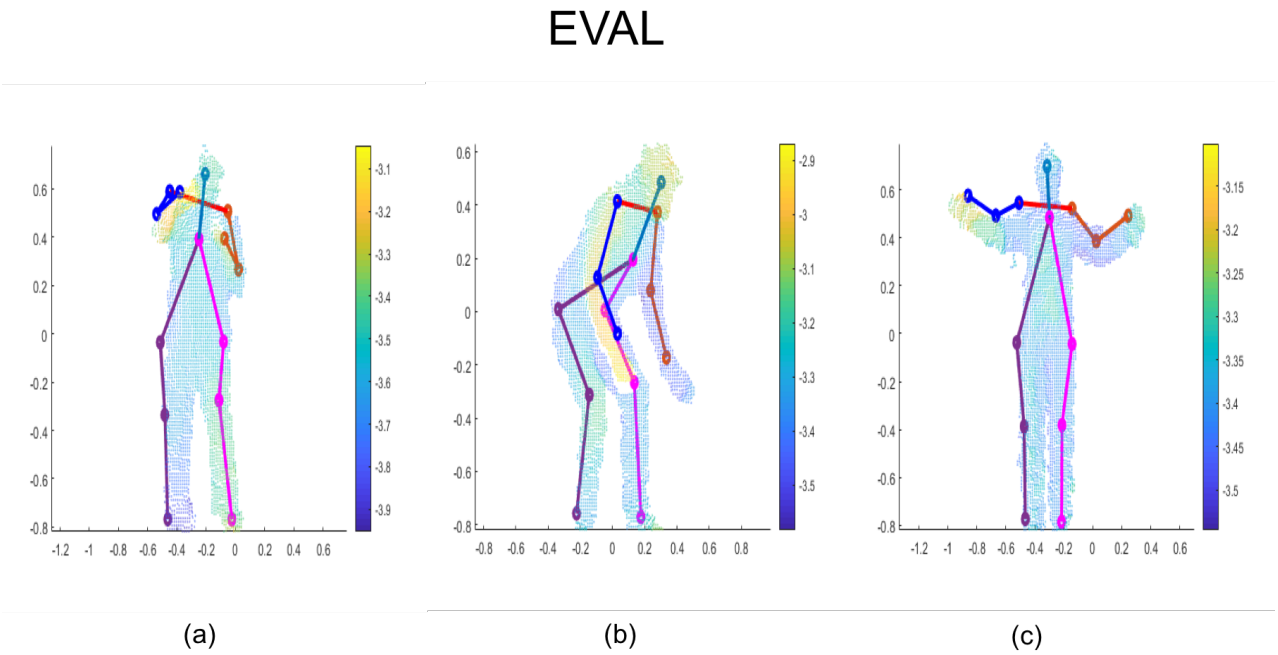

Figure 4.5: An example of EVAL dataset. The color bar shows the distance between the point cloud and camera. Among figures, Figure(a) and (b) are males while figure (c) is a female. The motions in the figure are boxing, sitting down, and hand waving.

4.4 Summary

Two commonly used datasets, ITOP and EVAL, are introduced and evaluation metrics are also presented in this chapter. Both datasets include the complex motions, depth images and corresponding joint data. The main evaluation metric is mean Average Precision, which is used to measure results between the predicted position and ground truth. The experimental results on these datasets and evaluation metrics are provided in next chapter.

Chapter 5

Experiments

In this chapter, we evaluated our proposed approach on two public datasets: EVAL [10] and ITOP [14] 3D human motion dataset. Depending on the different datasets, we estimated the different number of key body joints. The key body joints from the ITOP dataset is 15, while the key body joints from the EVAL dataset is 14. The details of data preparation are provided in Section 5.1. The training parameters for our model on both ITOP and EVAL dataset are presented in section 5.2.1. Also, we show the effects of our two-stage training strategy in section 5.2.2. Additionally, our experiment results are analyzed in both section 5.2.3 and section 5.2.4. Finally, in section 5.3, we compare our method with the state-of-the-art methods which are random forest(RF)[44], random

tree walk algorithm (RTW) [24], iterative error feedback (IEF) [6], viewpoint-invariant feature-based method (VI) [14] on both ITOP and EVAL datasets. Next, we discuss the most challenging joints and how our approach may be influenced by the different tasks performed from the user.

5.1 Data preparation

The depth maps with its corresponding joint data in ITOP and EVAL 3D human pose dataset are randomly shuffled, respectively. 5-fold cross validation is used in experiments. To produce the most convincing test results and verify the generalization ability of our model, all training and testing data are divided by the subjects. For the ITOP dataset, there are 17k and 4k depth data separately for training and testing after we remove invalid data. About 3k of the training data is used for validation. For the EVAL dataset, most of the invalid data is lack of Vicon marker positions. We remove these invalid data, and keep $8,158$ of front-facing depth images. One subject is selected for the test (about 2k frames) and the other two subjects are selected for training (about 5k frames). Besides, there are 800 frames for validation. Since the joint positions are made up of Vicon marker positions, we calculate the average position of markers for each joint.

It is the first step for our experiments to convert the depth map into point cloud. The process that project the image point to the world coordinate point has been shown in Figure 5.1. In a 2D image, each coordinate (U, V) has a vector to represent the color of the point. Similarly, the pixel value of each coordinate (U, V) from the depth map is a floating-point number. This number named depth value indicates the distance from the measurement plane. As shown in Figure 5.1, three-dimensional space is represented by the (X, Y, Z) coordinate system with the origin point (camera center). Z represents the depth value, while X and Y respectively represent the horizontal and vertical value. The process of projection is a transformation from the image coordinate system to the world coordinate system. The constraint of the transformation is the camera internal

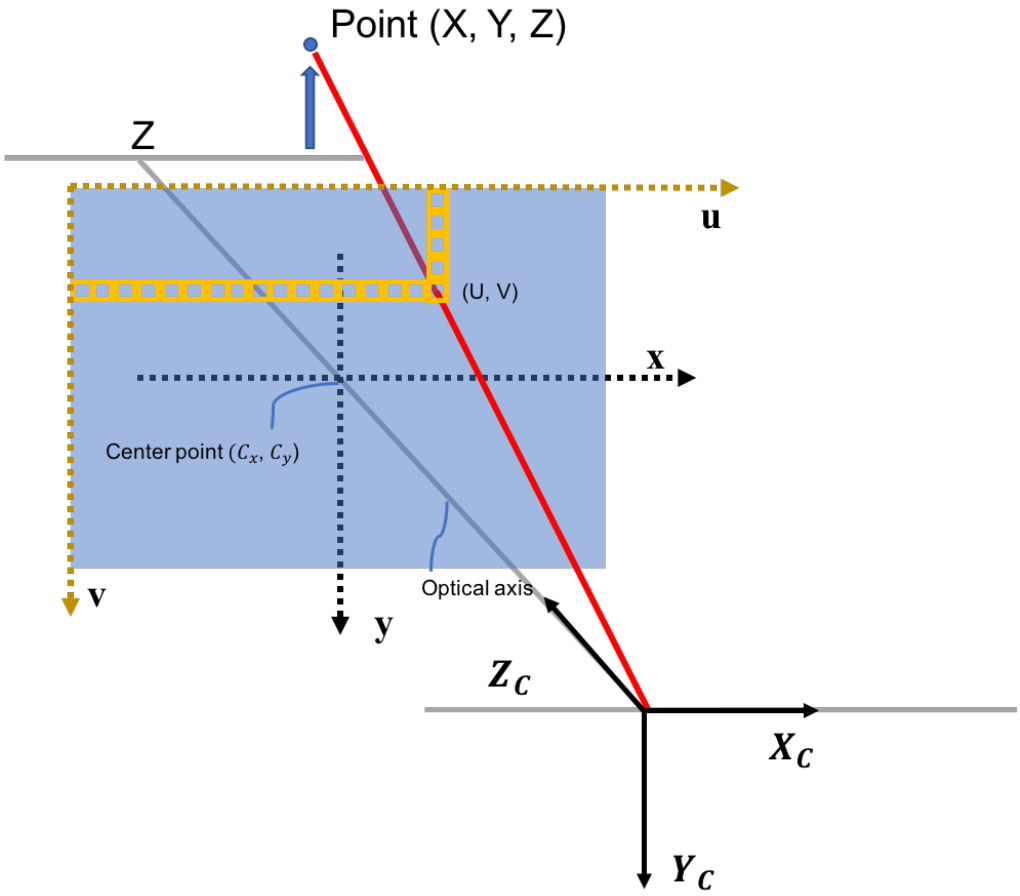

Figure 5.1: The process of projecting the image point (U, V) to the world coordinate point (X, Y, Z). The direction of vector Z_c is the optical axis, while the center point is (C_x, C_y) on the path of light propagation. Both image coordinate system $(u, v)$and world coordinate system (X_c, Y_c, Z_c) are shown in this figure. The yellow rectangle boxes represent the pixels of the image.

parameter. Since the point cloud is a collection of points in three-dimensional space, these points have the same reference coordinate system. Equation 5.1 is the formula of transformation.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{D} \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{5.1}$$

where f_x and f_y is the focal length in the direction of x and y, and the unit is pixel. (C_x, C_y) is the center point in image coordinate systems. X_c, Y_c, Z_c are point cloud coordinate systems; D is the depth value from the depth map. If there are the intrinsic and extrinsic parameters of the camera, the 2D coordinate system can be projected back to the 3D coordinate system.

5.2 Experiments on ITOP and EVAL dataset

In this section, the experimental results on ITOP and EVAL 3D human motion datasets are presented to provide comprehensive insight into the performance of our overall proposed model.

5.2.1 Experiment setup

All experiments were performed on a computer with one Intel Core i7-9700K CPU, dual Nvidia GTX1080ti GPUs with a total of 24GB memory, and 64GB of RAM. The operating system is Ubuntu 16.04. The human pose network and preprocessing were implemented by the TensorFlow framework and Point Cloud Library (PCL) [42] with CUDA 9.1. All depth maps from two public datasets are converted into point clouds. After preprocessing mentioned in section 3.1, the background objects are removed from the point clouds. The input to our pose net is a point cloud and outputs are the 3D positions of M human key body joints. There are two steps in our training process. The optimizations for both training steps are Adam [26]. For the first step to train the

spatial transform network, the initial learning rate is 0.001, and decay rates (β_1, β_2) are separately 0.9 and 0.999. Epsilon of Adma is $1e-08$. The learning rate is divided by 10 after 50 epochs. After about 80 epochs, we stop the first training and start the second training step. For the second step of training, the initial learning rate is 0.0001. Regularization strength is set to 0.0005. The number of neighbor points K is 16. All MLP and fully connected layers include the Relu activation function, except the last layer. We train the network with a batch size of 4. For a more in-depth training of spatial transform network, we randomly sampled the data in each training step. When we make the evaluation, the estimated 3D key body joint locations are reconstructed from the network outputs:

$$J = \digamma(X_t^{nor}, \omega^*) \begin{bmatrix} b_w & 0 & 0 \\ 0 & b_h & 0 \\ 0 & 0 & 1 \end{bmatrix} + b_c \tag{5.2}$$

where b_h, b_w, and b_c are person's height, width, and center separately; X_t^{nor} is the point cloud after normalization and segmentation; ω^* is the weights of the pre-trained network; $\digamma$ represents the our proposed network.

5.2.2 Effects of Two-stage Training Strategy on Spatial Transform Network

The main objective of a spatial transform network is to correct this disordered point cloud and make it into an ordered spatial structure. Therefore, the effective use of the spatial transform network is a significant part for our experiment. Meanwhile, a two-stage training strategy was proposed for training the spatial transform network. There are two experiments conducted on both ITOP and EVAL dataset to show the effectiveness of applying a training strategy based on transfer learning. We compare the results of the non-two-stage training strategy on the spatial transform network and that adopting two-stage training strategy. At first, we train the whole network to achieve the best state of the spatial transform network and randomly arrange the points from the point

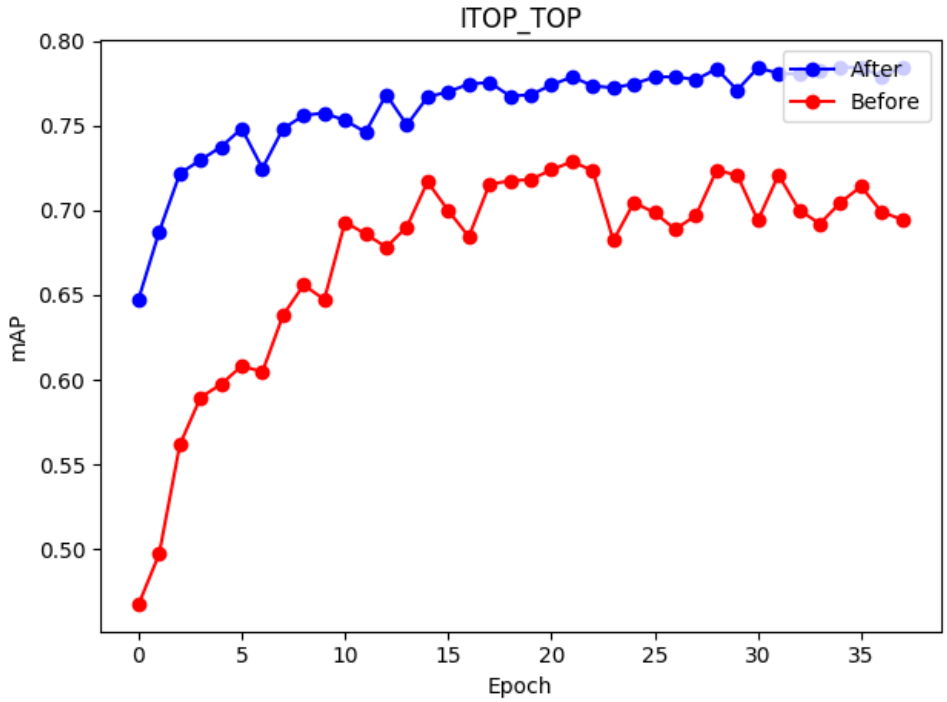

Figure 5.2: The mAP results after and before the adoption of our two-stage training strategy on ITOP top-view dataset

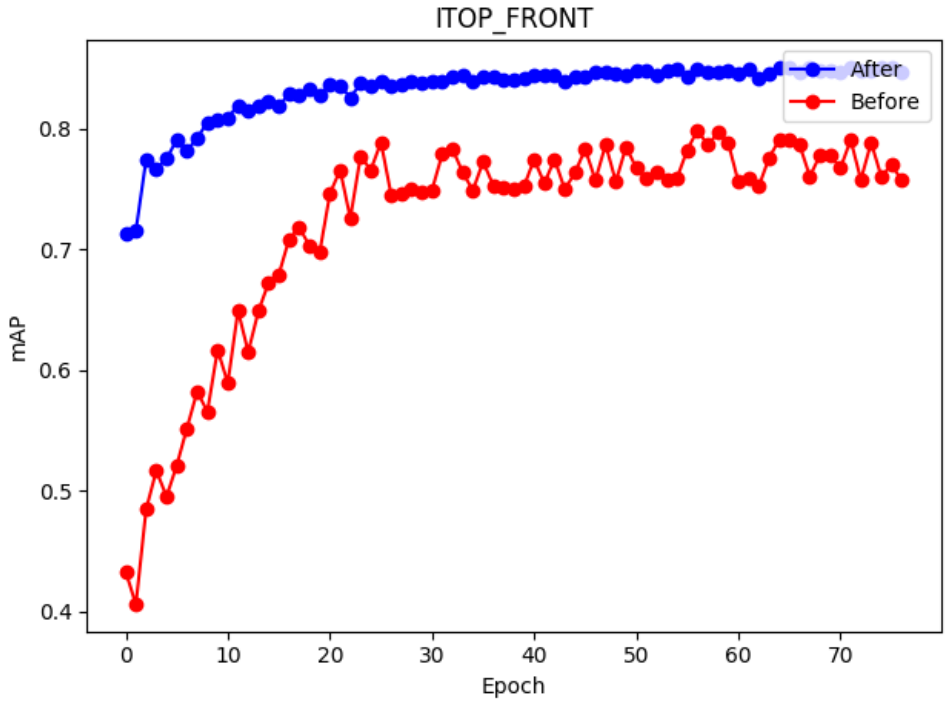

Figure 5.3: The mAP results after and before the adoption of our two-stage training strategy on ITOP front-view dataset

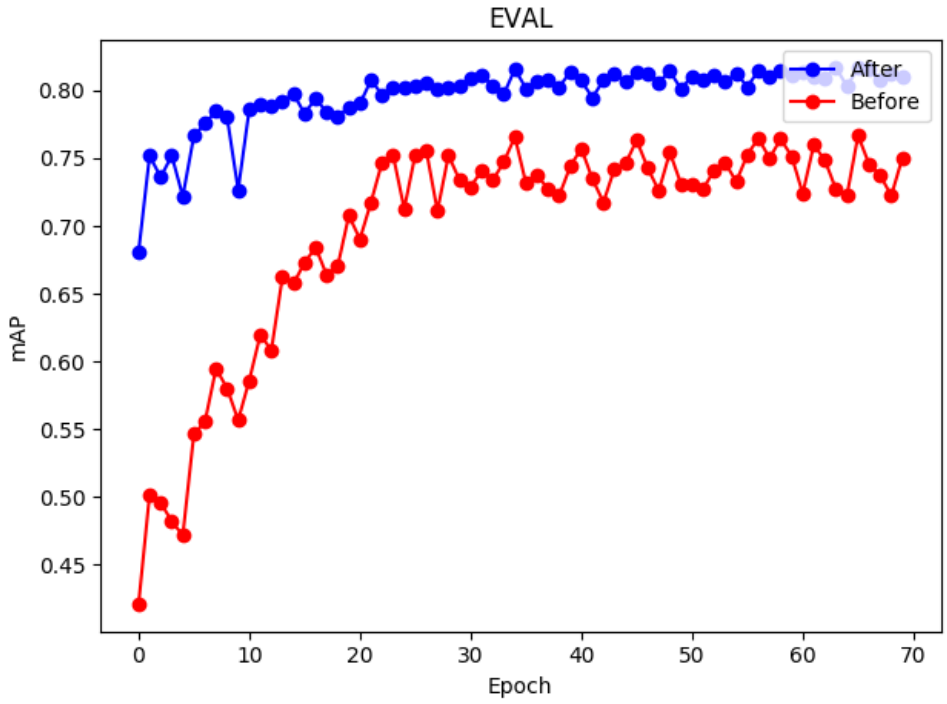

Figure 5.4: The mAP results after and before the adoption of our two-stage training strategy on EVAL dataset

cloud. The next step is that we use the pre-trained model to feed the weights of the spatial transform network into the new model. After freezing the weights of the spatial transform network, we train the remaining model without randomly sampling.

Table 5.1: The Illustration of the effectiveness of our transferring learning strategy for both EVAL and ITOP dataset

Dataset	After two-stage training	Before two-stage training	Improvement
ITOP Front-view	85.11%	80.12%	4.99%
ITOP Top-view	78.46%	73.89%	4.57%
EVAL	80.11%	76.51%	3.60%

The mean average precision curves from ITOP and EVAL datasets are shown in Figure 5.2, 5.3, and 5.4 separately. These figures are composed of mean average precision (mAP) with its corresponding training epoch. The red line represents the estimation accuracy (mAP) before adopting the transfer learning on spatial transform network, while the blue line represents the estimation accuracy after the adoption. Among these figures, the results are improved into a higher value by the adoption of our two-stage training strategy. By using a pre-trained model, we speed up the training process of our network. Furthermore, we also listed Table 5.1 to see the improvement space more clearly. The values in the table refer to mean average precision (mAP). Our method has improved greatly on all datasets, from 3.6% to about 5%. Among them, the biggest improvement is ITOP front-view dataset, while the smallest improvement is EVAL dataset. The mAP is improved from 80.12 to 85.11 after applying the two-stage training strategy on ITOP front-view dataset. For the other part of ITOP, the mAP is also upgraded to 78.46. In the experiment of EVAL dataset, the improvement is more obvious. It is promoted into 80.11. The adoption of the transfer learning strategy can improve the results up to five percentages. The above data proves that our method can indeed improve the overall estimation.

5.2.3 Results and Analysis for ITOP dataset

As we mentioned in Chapter 4, there are two parts in the ITOP dataset. We train and evaluate our model on the front and top view of the dataset separately. The results are summarized in Table 5.2 by using the evaluation metrics mentioned in Chapter 4. We evaluated the 15 key joints of the human body, such as head, neck, shoulders, elbows, hands, torso, hips, knees, feet. Also, a mean mAP of all the human key joints is shown in the last row of the table. The total loss for training and testing on two parts of the ITOP dataset is shown in Figure 5.5 and Figure 5.6. We determine the training level of the model through loss curves. Our training model on both parts with the ITOP dataset has achieved a good performance based on these loss curves. Besides, the qualitative visualization results of our pose net on the ITOP dataset are shown in Figure 5.7 and Figure 5.8. In each figure, the left is the point cloud without background objects, while the right the corresponding key joints. In addition, the color bar in each frame shows a distance from the points to the depth camera.

Table 5.2: The experiment results for ITOP dataset

View	**Head**	**Neck**	**Shoulders**	**Elbows**	**Hands**	**Torso**	**Hips**	**Knees**	**Feet**	**Mean**
Front View	96.73	98.05	94.38	73.67	54.95	98.35	91.77	90.74	86.30	85.11
Top View	96.13	97.61	93.08	70.83	48.41	95.58	84.50	79.19	67.76	78.46

In both parts of the ITOP dataset, the prediction results of elbows, hands, and feet are lower than the mAP of 90, especially the estimation of the hand is not ideal. It is difficult to predict the hand joints because hands occupy a relatively small proportion in the body part. Our pose net performed well on the prediction of most main human body parts, which are head, neck/chest, shoulders, and torso, because these main parts provide the most rich depth information. After comparing the front view and the top view, the accuracy from lower body parts in the top-view part is lower than those in front-view part. Most of the lower body parts, such as hips, knees, and feet, are not

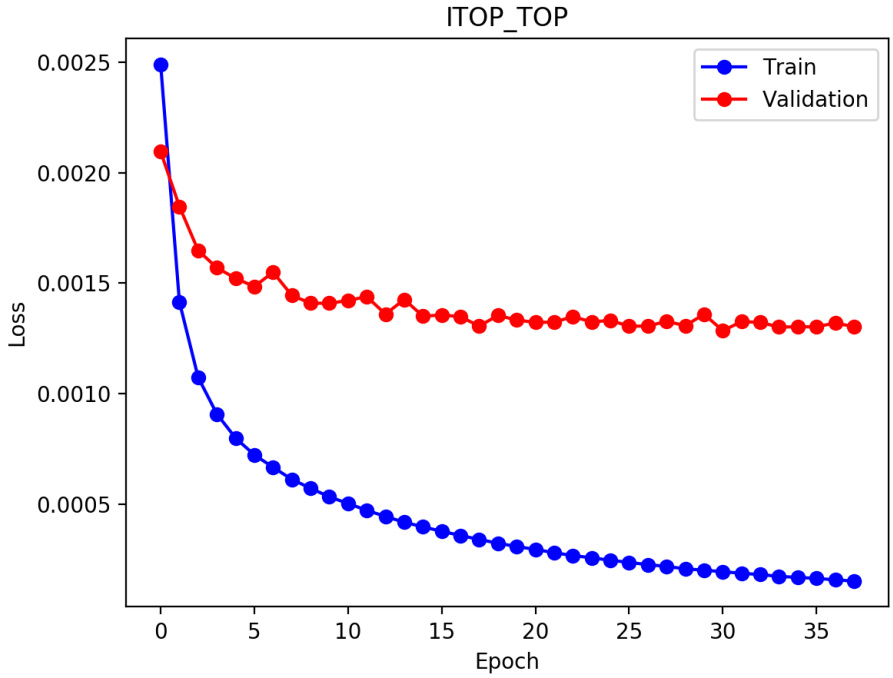

Figure 5.5: The total loss during training and testing for ITOP top-view dataset.

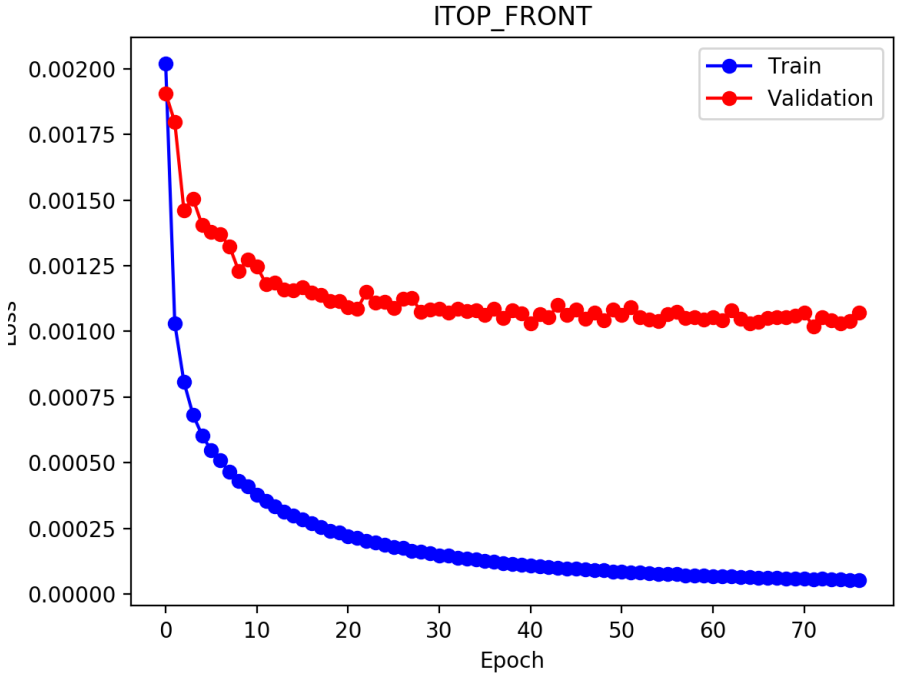

Figure 5.6: The total loss during training and testing for ITOP front-view dataset.

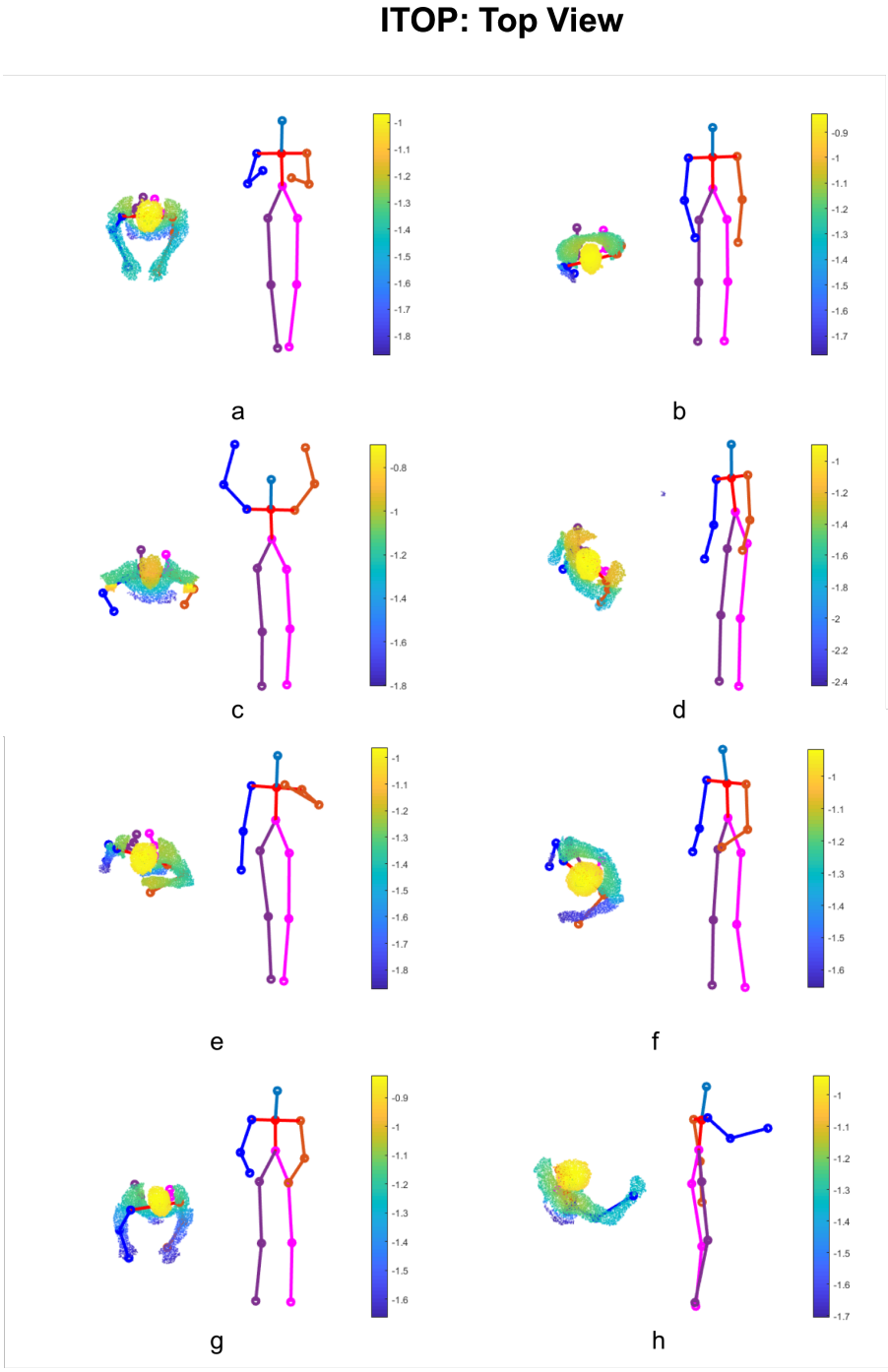

Figure 5.7: Qualitative results of our pose net on ITOP top-view dataset. The motions in the figure include raising hands, boxing, swinging, and standing.

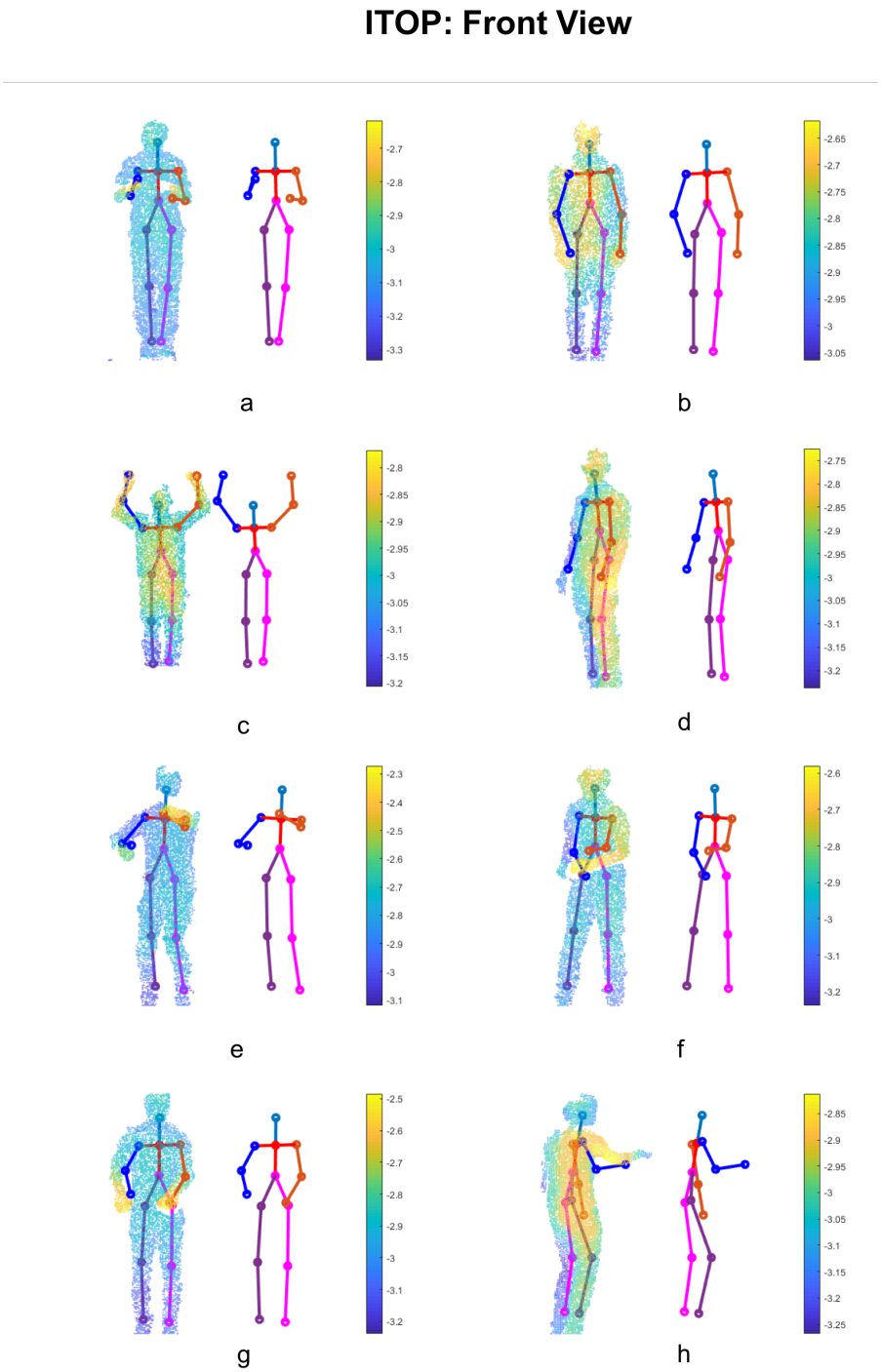

Figure 5.8: Qualitative results of our pose net on ITOP front-view dataset. The motions in the figure include raising hands, boxing, swinging, and standing.

visualized in the view from the top to the bottom as shown in Figure 5.7. This results in the lower mAP results when testing the top view part.

5.2.4 Results and Analysis for EVAL dataset

The second dataset to be evaluated is the EVAL dataset, which has 14 key body joints, such as head, chest, shoulders, elbows, hands, hips, knees, feet, and so on. The evaluation value for each point is listed in Table 5.3. Also, a mean mAP of all the human key joints is shown in the last row of the table. The total loss for training and testing on the EVAL dataset is shown in Figure 5.9. Our training model has achieved a good performance based on the loss curves. Loss is already in a state of convergence. Besides, the qualitative visualization results of our pose net on the EVAL dataset are shown in Figure 5.10. In each frame, the left is the point cloud without background objects, while the right is the corresponding key joints. The color bar in each frame shows a distance from the points to the depth camera.

Table 5.3: The experiment results for EVAL dataset

Dataset	**Head**	**Chest**	**Shoulders**	**Elbows**	**Hands**	**Hips**	**Knees**	**Feet**	**Mean**
EVAL	88.93	96.87	86.14	75.11	63.07	83.20	82.68	82.94	80.86

As shown in Table 5.3, the mean mAP of human body key joints is lower compared to the ITOP dataset. Similarly, the score in predicting the position of hands is not ideal. There is a large space for improvement in predicting the 3D position of hands. The predictions of other parts are basically around an mAP of 80. It is more difficult to make a pose estimation on the EVAL dataset than on the ITOP dataset. The motions in the ITOP dataset are simple, while motion in the EVAL dataset contains a large range of movements, such as rolling and kicking from different angles.

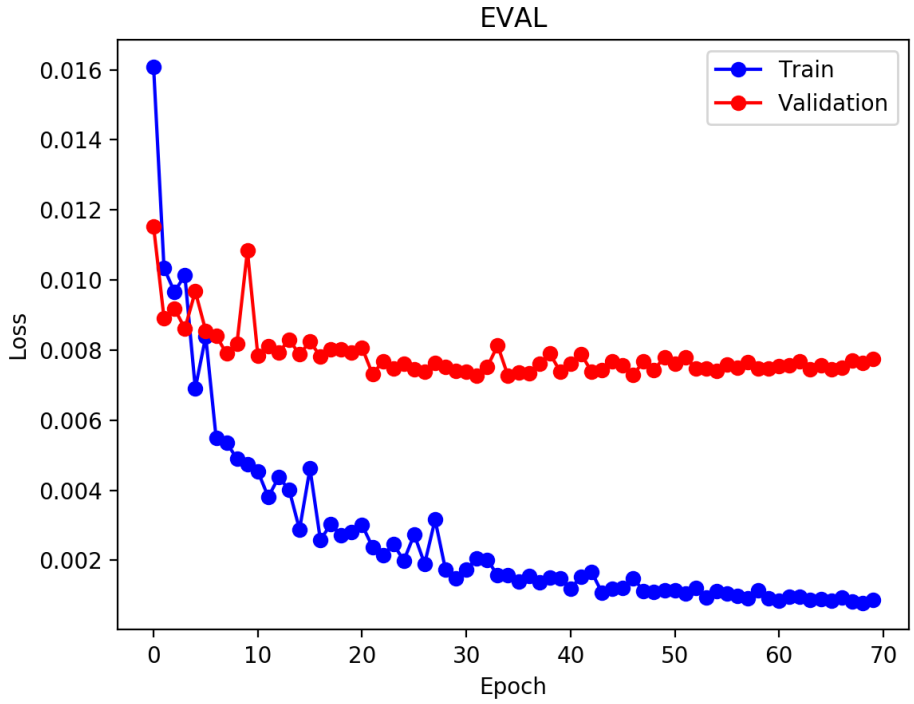

Figure 5.9: The total loss during training and testing for ITOP top-view dataset.

5.3 Comparison among the state-of-art methods

In this section, we compare our proposed approach with state-of-the-arts methods, which are random forest(RF)[44], random tree walk algorithm (RTW) [24], iterative error feedback (IEF) [6], viewpoint-invariant feature-based method (VI) [14] on the ITOP dataset in section 5.3.1 and 5.3.2. Next, we compare our approach with RTW and VI in section 5.3.3. Simultaneously, we analyze the performance between state-of-the-art methods and our methods through experimental results. For both ITOP and EVAL dataset, the mAP of the upper body and lower body has been provided in Table 5.4, 5.5, and 5.6. Table 5.4 and 5.5 represent the results of the ITOP front-view and top-view dataset separately, while Table 5.6 shows the results of the EVAL dataset. The upper part of the body includes head, neck/chest, shoulders, elbows, and hands, while the lower part of the body includes the hips, knees, and feet.

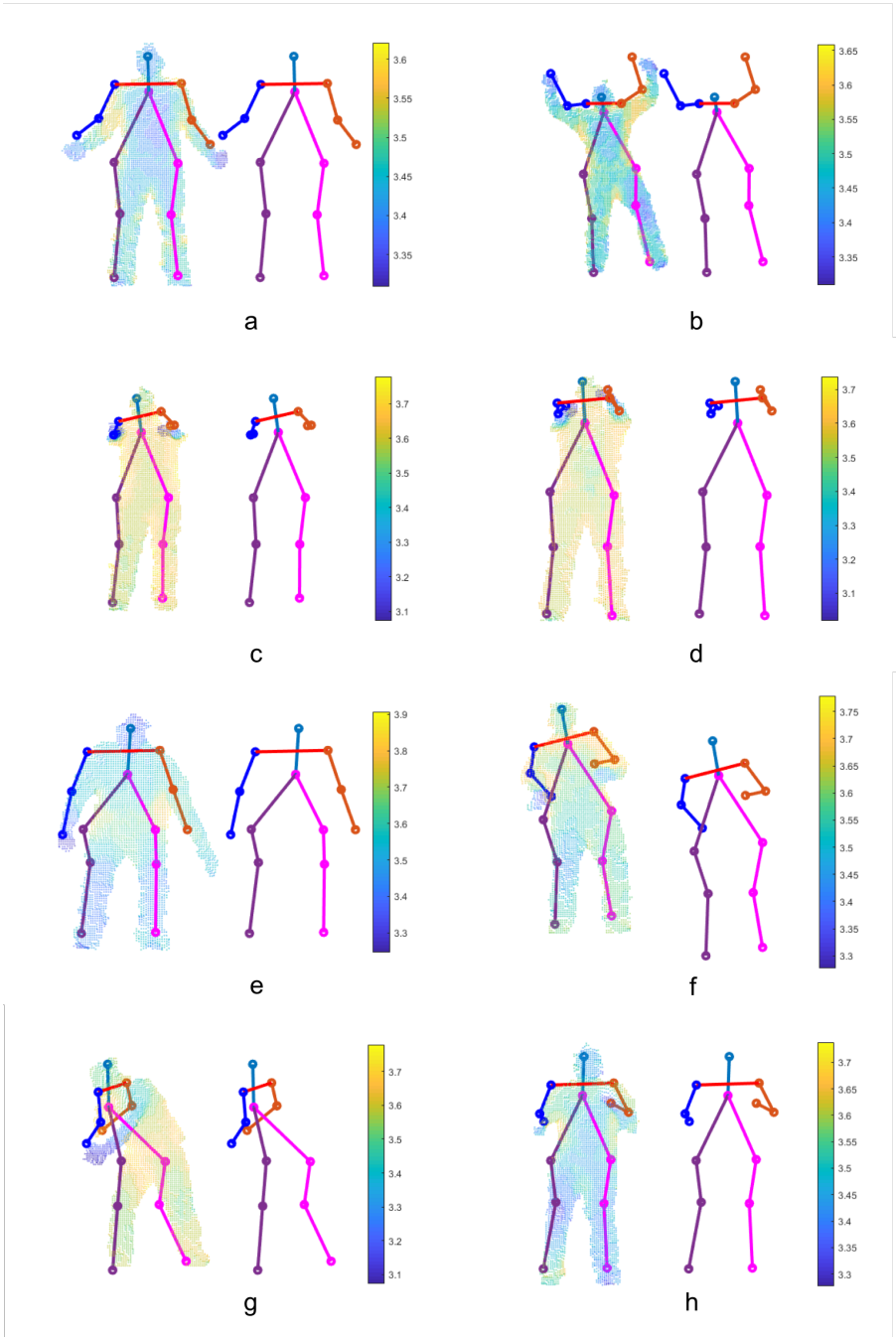

Figure 5.10: Qualitative results of our pose net on the EVAL dataset. The motions in the figure include raising handswaving hands, squatting down, and swinging.

5.3.1 ITOP top-view dataset

For the top-view of the ITOP dataset, the view angle of the collected dataset results in a considerable loss of depth information. The top-view of the ITOP dataset suffers serious depth information loss. Figure 5.7 can be taken as a representative example of such a situation. Most of the depth information is invisible, especially the parts of the lower body. This is also the most difficult task in estimating the human key body joints on the top-view part of the ITOP dataset. Moreover, compared with other state-of-the-art approaches, our approaches have a pretty good performance.

Table 5.4: Comparison of the proposed method with state-of-the-art methods on the ITOP (top-view) dataset

Dataset	**ITOP(top-view)**				
	RF[44]	**RTW** [24]	**IEF** [6]	**VI** [14]	**Ours**
Head	95.4	**98.4**	83.8	98.1	96.13
Neck	**98.5**	82.2	50.0	97.6	97.61
Shoulders	89.0	91.8	67.3	**96.1**	93.08
Elbows	57.4	80.1	40.2	**86.2**	70.83
Hands	49.1	76.9	39.0	**85.5**	48.41
Torso	80.5	68.2	30.5	72.9	**95.58**
Hips	20.0	55.7	38.9	61.2	**84.50**
Knees	2.6	53.9	54.0	51.6	**79.19**
Feet	0.0	28.7	62.4	51.5	**67.76**
Upper Body	73.1	84.8	51.7	**91.4**	77.30
Lower Body	7.5	46.1	53.3	54.7	**77.15**
Mean	47.4	68.2	51.2	75.5	**78.46**

From Table 5.4, although VI has the best score in the assessment of the upper body

with the mAP of 91.4, our human pose net has a score with the mAP of 77.15, which is higher than other approaches. On estimating the whole body joints, our approach has a good mAP score performance of 78.46, which is higher than the second-ranking method VI. In predicting the key points of the lower body, the result of our proposed pose net on estimating each part can exceed the score of at least 16% more than the other approaches. Especially, the score of estimation on the torso is an mAP of 95.58%, which is far higher than other approaches. However, the estimation accuracy of the upper body part equals the lower part, and the main difference exists in the prediction of the hand part.

5.3.2 ITOP front-view dataset

The results of upper and lower body joints between the state-of-the-arts approaches and our proposed pose net are listed in Table 5.5. On estimating the upper parts of the body, RTW has the best score with an mAP of 84.8%. The estimated results of other upper parts from our approach are the same as the state-of-the-art approaches, espeically the neck joint with a high score of 98.05% mAP. Our method has a high score of 98.05% mAP on the estimation of the neck joint. The score of RF, RTW, IEF, and VI are lower than ours in predicting feet and knees. Thus, the result on estimating lower body parts can exceed the score of 20% more than the other approaches. From Table 5.5, our method performs better in predicting the whole body with an mAP of 85.11% than others.

5.3.3 EVAL dataset

In the EVAL dataset, we compare our method with RTW and VI. Random forest depends on a per-pixel body part labeling, which is not provided by EVAl dataset. Therefore, RF and IEF are not considered in this section. Due to the application of the Vicon system in this dataset, the accuracy of the 3D joint position from EVAL dataset is very high. While both ITOP and EVAL collected data from the angle of the front view, EVAL dataset has more complex motion than ITOP. The results of EVAL dataset are relatively lower

Table 5.5: Comparison of the proposed method with state-of-the-art methods on the ITOP (front-view) dataset

Dataset	**ITOP(front-view)**				
Body part	**RF**[44]	**RTW** [24]	**IEF** [6]	**VI** [14]	**Ours**
Head	63.8	97.8	96.2	**98.1**	96.73
Neck	86.4	95.8	85.2	97.5	**98.05**
Shoulders	83.3	94.1	77.2	**96.5**	94.38
Elbows	73.2	**77.9**	45.4	73.3	73.67
Hands	51.3	**70.5**	30.9	68.7	54.95
Torso	65.0	93.8	84.7	85.6	**98.35**
Hips	50.8	80.3	83.5	72.0	**91.77**
Knees	65.7	68.8	81.8	69.0	**90.74**
Feet	61.3	68.4	80.9	60.8	**86.30**
Upper Body	70.7	**84.8**	61.0	84.0	80.10
Lower Body	59.3	72.5	82.1	67.3	**89.60**
Mean	65.8	80.5	71.0	77.4	**85.11**

than the ITOP front-view dataset. However, in comparison with two other approaches, our proposed human net has achieved better results on the EVAL dataset.

As shown in Table 5.6, our estimation results of the hand and elbows joints are far more than the other two methods. Also, comparing the results estimated on the chest, our approach can reach an mAP of 96.87%, higher than the other two approaches. Therefore, we rank first in estimating the upper body parts. VI achieved a higher score on the evaluation of the lower body with a mAP of 89.2. For the whole body, our approach was the first one with the whole body mAP of 80.86, which was much higher than the others (RTW and VI). This comes from the accurate estimations in the hand and elbow joints.

Table 5.6: Comparison of the proposed method with state-of-the-art methods on the EVAL dataset

Dataset	**EVAL**		
Boday part	**RTW** [24]	**VI**[14]	**Ours**
Head	90.9	**93.9**	88.93
Chest	87.4	94.7	**96.87**
Shoulders	**87.8**	87.0	86.14
Elbows	27.5	45.5	**75.11**
Hands	32.3	39.6	**63.07**
Torso	–	–	–
Hips	–	–	83.20
Knees	83.4	**86.0**	82.68
Feet	90.0	**92.3**	82.94
Upper Body	59.2	73.8	**79.31**
Lower Body	86.7	**89.2**	82.94
Mean	68.3	74.1	**80.86**

5.3.4 Discussion

In our experiment, the most challenging joints are hand and elbow. The mean average precision from both datasets are below 75. The main reason is the lack of enough depth information. The motion range of both of the two joints is large, leading that the two parts beyond the capturing scope of the camera. Moreover, these two joints are easily occluded by other body parts from the view of the camera. Once these two parts are partially occluded, it results in that the extracted body parts in the point cloud image are not connected with each other and cause the depth information to be filtered out in the processing of segmentation.

For the top-view part of ITOP dataset, VI performs much better than ours on predicting the joints from upper body parts. The main difference is the results of hands and elbows since the these two parts are lack of enough depth information. Compared to the top-view part, our approach performs much better when estimating pose from front-view part. Main contributions are from lower body parts. The data of these two parts are collected from different angles. Due to the angle problem, many parts of the body are blocked in top-view part. This lead to the lower accuracy. For the EVAL dataset, VI performs a bit better than ours on predicting the joints from lower body parts. The main difference is foot part. Since the feet are too close to the ground, this part can be easily been segment from the main human body part. Therefore, developing an efficient segmentation method is the next step work for our 3D pose estimation.

The application scenarios of our proposed approach are wide, including rehabilitation training, exercise coaching, sport player tracking, etc. In the rehabilitation training, most of the environments are controlled structured environments and the designed training is typically a slow tracking movement for a single patient. In this situation, our proposed approach well suits the task. In the exercise coaching, the movement can be fast which needs our proposed approach to be efficient enough to track the movement. In this case, we can tailor the model to be a light-weight simplified one by eliminating the neural links with small weights and increasing the number of sharing weights between clustered links. In the sport player tracking, the tasks are typically multi-player tracking which often involves occlusions. To overcome the occlusion issue, we can use maximum likelihood principle or posterior estimation to make correct joint association and pose estimation by fully utilizing the prior knowledge of the constraint between adjacent joints. More specifically, the occlusion can be modeled as a crossing of two segments based on computational geometry principle. Different combinations of movement sequences are then compared in the sense of probability by taking human movement restrictions into account [57].

5.4 Summary

In this chapter, we have conducted experiments on two commonly used datasets and verified the effectiveness of our two-stage strategy on training the spatial transform network. The experimental results compared to the-state-of-the-art approaches have shown that our approach achieves the best performance. In addition, we provide the discussions for the low accuracy from some body and point out the direction for our further improvement.

Chapter 6

Conclusions and Future Work

To the best of our knowledge, we propose a human pose net that estimates human pose directly from point cloud with a single depth map for the first time. 3D point clouds are converted from 2.5 depth maps. As the great performance of Dynamic Graph CNN network [54] and PointNet [38], [39] in field of segmentation and classification based on point clouds, we conduct pose estimation based on the modified Dynamic Graph CNN network and PointNet to regress the 3D positions of joints directly. To handle the influence of background objects, we segment and resample the point cloud of the human body by using the Euclidean cluster extraction. Also, the operation of segmentation and sampling reduces the number of point clouds. The point cloud is fed into our modified

Dynamic Graph CNN network after being normalized by its height and width. Moreover, to enhance the performance of our pose net, we adopt a two-stage training strategy and verified the effectiveness of this strategy on training the spatial transform network.

Besides, the experiments and comparison of our proposed network to existing approaches are conducted on two available public datasets, ITOP and EVAL 3D human pose dataset. The mean average precision is determined as the accuracy of our estimation in the evaluation. In the experiments, we compare not only the accuracy of each human key joints but also the accuracy of the upper and lower body. Overall, the performance of our proposed method on two depth-based datasets, ITOP and EVAL, obtains results superior to the state-of-the-art methods. The result illustrates that EVAL is more challenging than ITOP since the motions in this dataset are relatively complex.

However, putting aside what we have successfully achieved, several critical issues can be addressed for further improvements. First, estimating a more accurate position on the small part of the body using the current depth information is the first improvement for our proposed network. The results of all the approaches we list in the experiment are generally lower than an mAP of 80 for the prediction of hands and elbows. Improving the estimation of these parts can have a positive impact on the whole results. Also, a fast real-time model needs to be developed. Although machine learning, such as random forest, performs a little worse, it has a huge ability on fast pose estimation. Reducing the structure of the network or compressing model to cut down the running time is a new field on the deep learning method.

References

[1] Mykhaylo Andriluka et al. "2D Human Pose Estimation: New Benchmark and State of the Art Analysis". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[2] Mykhaylo Andriluka et al. "2D Human Pose Estimation: New Benchmark and State of the Art Analysis". In: *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3686–3693.

[3] Sapp Benjamin and Taskar Ben. "MODEC: Multimodal Decomposable Models for Human Pose Estimation". In: *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3674–3681.

[4] Yiheng Cai, Xueyan Wang, and Xinran Kong. "3D Human Pose Estimation from RGB+D Images with Convolutional Neural Networks". In: *proceeding of the International Conference on Biomedical Engineering and Bioinformatics*. 2018.

[5] Zhe Cao et al. "Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields". In: *Proceeding of European Conference on Computer Vision*. 2017, pp. 7291–7299.

[6] João Carreira et al. "Human pose estimation with iterative error feedback". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4733–4742.

[7] Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. "A Real-Time Human Action Recognition System Using Depth and Inertial Sensor Fusion". In: *IEEE Sensors Journal* 16.3 (2016), pp. 773–781.

[8] Phuong Minh Chu et al. “Fast point cloud segmentation based on flood-fill algorithm”. In: *proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent System*. 2017.

[9] Jia Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *Proceeding of IEEE Computer Vision and Pattern Recognition*. 2009.

[10] Varun Ganapathi et al. “Real-Time Human Pose Tracking from Range Data”. In: *proceedings of the European Conference on Computer Vision*. 2012, pp. 738–751.

[11] Liuhao Ge1 et al. “Hand PointNet: 3D Hand Pose Estimation using Point Sets”. In: *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8417–8426.

[12] Daniel Grest, Jan Woetzel, and Reinhard Koch. *Nonlinear Body Pose Estimation from Depth Images*. Springer, 2005.

[13] Hengkai Guo et al. “Towards goodpractices for deep 3d hand pose estimation”. In: *proceedings of the IEEE International Conference on Image Processing*. 2017, pp. 4512–4516.

[14] Albert Haque et al. “Towards Viewpoint Invariant 3D Human Pose Estimation”. In: *proceedings of the European Conference on Computer Vision*. 2016, pp. 160–177.

[15] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[16] Kaiming He et al. “Mask R-CNN”. In: *Proceeding of IEEE International Conference on Computer Vision*. 2017, pp. 2980–2988.

[17] Li He et al. “Depth-images-based pose estimation using regression forests and graphical models”. In: *Neurocomputing* 164 (2015), pp. 210–219.

[18] Thomas Helten et al. “Personalization and Evaluation of a Real-Time Depth-Based Full Body Tracker”. In: *proceedings of the International Conference on 3D Vision*. 2013, pp. 279–286.

[19] Thomas Helten et al. “Sensor fusion for 3D human body tracking with an articulated 3d body model”. In: *proceedings of the IEEE International Conference on Robotics and Automation*. 2006.

[20] Andrew G. Howard et al. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. In: (2017). arXiv: 1704.04861.

[21] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2261–2269.

[22] Catalin Ionescu et al. “Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2014), pp. 1325–1339.

[23] Sam Johnson and Mark Everingham. “Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation”. In: *proceedings of the British Machine Vision Conference*. 2010.

[24] Ho Yub Jung et al. “Random tree walk toward instantaneous 3d human pose estimation”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2467–2474.

[25] Lipeng Ke1 et al. “Multi-Scale Structure-Aware Network forHuman Pose Estimation”. In: *proceedings of the European Conference on Computer Vision*. 2018, pp. 731–746.

[26] Diederik P. Kingma and Jimmy Lei Ba. “Adam: A Method for Stochastic Optimization”. In: *proceedings of the International Conference for Learning Representations*. 2015.

[27] Hema S Koppula, Rudhir Gupta, and Ashutosh Saxena. “Learning Human Activities and Object Affordances from RGB-D Videos”. In: *International Journal of Robotics Research* (2017).

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *proceedings of the Advances in Neural Information Processing Systems*. 2012.

[29] Yann Lecun, Leon Bottou, and Yoshua Bengio. "Gradient-based learning applied to document recognition". In: *proceedings of the IEEE*. Vol. 86. 11. 1998, pp. 2278–2324.

[30] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *proceedings of the European Conference on Computer Vision*. 2014, pp. 740–755.

[31] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Proceeding of European Conference on Computer Vision*. 2016, pp. 21–37.

[32] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: *proceeding of International Conference on Machine Learning*. 2013.

[33] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. "V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation from a Single Depth Map". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5079–5088.

[34] Balazs Nagy and Csaba Benedek. "3D CNN-Based Semantic Labeling Approach for Mobile Laser Scanning Data". In: *IEEE Sensors Journal* 19.9 (2019), pp. 10034–10045.

[35] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked Hourglass Networks forHuman Pose Estimation". In: *proceedings of the European Conference on Computer Vision*. 2016, pp. 483–499.

[36] Tomas Pfister, James Charles, and Andrew Zisserman. "Flowing ConvNets for Human Pose Estimation in Videos". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1913–1921.

[37] Kohli Pushmeet and Shotton Jamie. *Key Developments in Human Pose Estimation for Kinect*. Consumer Depth Cameras for Computer Vision. Springer, Jan. 2013.

[38] Charles R. Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 77–85.

[39] Charles R. Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *proceedings of the Advances in Neural Information Processing Systems*. 2017, pp. 5099–5108.

[40] Varun Ramakrishna et al. "Pose Machines: Articulated Pose Estimationvia Inference Machines". In: *proceedings of the European Conference on Computer Vision*. 2014.

[41] Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. "Non-maximum Suppression for Object Detection by Passing Messages Between Windows". In: *Proceeding of Asian Conference on Computer Vision*. 2014, pp. 290–306.

[42] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *proceedings of the IEEE International Conference on Robotics and Automation*. 2011.

[43] Abdulmotaleb El Saddik. "Digital Twins: The Convergence of Multimedia Technologies". In: *IEEE MultiMedia 25.2*. 2018, pp. 87–92.

[44] Jamie Shotton et al. "Real-Time Human Pose Recognition in Parts from Single Depth Images". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2011, pp. 1297–1304.

[45] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *Proceeding of International Conference on Learning Representations*. 2015.

[46] Jaeyong Sung et al. “DeepPose: Human Pose Estimation via Deep Neural Networks”. In: *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1653–1660.

[47] Jaeyong Sung et al. “Unstructured Human Activity Detection from RGBD Images”. In: *Proceeding of International Conference on Robotics and Automation*. 2011.

[48] Christian Szegedy et al. “Going Deeper with Convolutions”. In: *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[49] Jonathan Tompson et al. “Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation”. In: 2014.

[50] Jonathan Tompson et al. “Joint Training of a Convolutional Network and aGraphical Model for Human Pose Estimation”. In: *Proceeding of Advances in Neural Information Processing Systems*. 2014.

[51] Alexander Toshev and Christian Szegedy. “DeepPose: Human Pose Estimation via Deep Neural Networks”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1653–1660.

[52] N. P. van der Aa et al. “UMPM benchmark: A multi-person dataset with synchronized video and motion capture data for evaluation of articulated human motion and interaction”. In: *IEEE International Conference on Computer Vision Workshops*. 2011, pp. 1264–1269.

[53] Min Wang et al. “DRPose3D: Depth Ranking in 3D Human Pose Estimation”. In: *proceedings of the International Joint Conference on Artificial Intelligence*. 2018, pp. 978–984.

[54] Yue Wang et al. “Dynamic Graph CNN for Learning on Point Clouds”. In: *ACM Transactions on Graphics* 1.1 (2019), pp. 1–13.

[55] Shih-En Wei et al. “Convolutional Pose Machines”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4724–4732.

[56] Bin Xiao, Haiping Wu, and Yichen Wei. “Simple Baselines for Human Pose Estimation and Tracking”. In: *Proceeding of European Conference on Computer Vision*. 2018, pp. 466–481.

[57] Bowen Yang, Haiwei Dong, and Abbulmotaleb El Saddik. “Development of a self-calibrated motion capture system by nonlinear trilateration of multiple Kinects v2”. In: *IEEE Sensors Journal* 17.8 (2017), pp. 2481–2491.

[58] Lin Yang et al. “Evaluating and improving the depth accuracy of Kinect for Windows v2”. In: *IEEE Sensors Journal* 15.8 (2015), pp. 4275–4284.

[59] Wei Yang et al. “3D Human Pose Estimation in the Wild by Adversarial Learning”. In: *proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5255–5264.

[60] Wei Yang et al. “Learning Feature Pyramids for Human Pose Estimation”. In: *proceedings of the IEEE Conference on Computer Vision*. 2017, pp. 1302–1310.

[61] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: *proceedings of the Advances in Neural Information Processing Systems*. 2014, pp. 3320–3328.

[62] Yuan Yu, Pradeep Kumar Gunda, and Michael Isard. “Distributed Aggregation for Data-Parallel Computing: Interfaces and Implementations”. In: *Proceeding of ACM Symposium on Operating Systems Principles*. 2009, pp. 247–260.

[63] Xiu Yuliang et al. “Pose Flow: Efficient Online Pose Tracking”. In: *Proceeding of British Machine Vision Conference*. 2018.

[64] Xiangyu Zhang et al. “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices”. In: *proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6848–6856.

[65] Xingyi Zhou et al. “Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach”. In: *proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 398–407.