# A Human Kinetic Dataset and A Hybrid Model for 3D Human Pose Estimation

by

Jianquan Wang

Thesis submitted to the University of Ottawa
In partial fulfillment of the requirements
For the MA.S.C. degree in
Electrical and Computer Engineering

uOttawa

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

# Abstract

Human pose estimation represents the skeleton of a person in color or depth images to improve a machine's understanding of human movement. 3D human pose estimation uses a three-dimensional skeleton to represent the human body posture, which is more stereoscopic than a two-dimensional skeleton. Therefore, 3D human pose estimation can enable machines to play a role in physical education and health recovery, reducing labor costs and the risk of disease transmission. However, the existing datasets for 3D pose estimation do not involve fast motions that would cause optical blur for a monocular camera but would allow the subjects' limbs to move in a more extensive range of angles. The existing models cannot guarantee both real-time performance and high accuracy, which are essential in physical education and health recovery applications. To improve real-time performance, researchers have tried to minimize the size of the model and have studied more efficient deployment methods. To improve accuracy, researchers have tried to use heat maps or point clouds to represent features, but this increases the difficulty of model deployment.

To address the lack of datasets that include fast movements and easy-to-deploy models, we present a human kinetic dataset called the Kivi dataset and a hybrid model that combines the benefits of a heat map-based model and an end-to-end model for 3D human pose estimation. We describe the process of data collection and cleaning in this thesis. Our proposed Kivi dataset contains large-scale movements of humans. In the dataset, 18 joint points represent the human skeleton. We collected data from 12 people, and each person performed 38 sets of actions. Therefore, each frame of data has a corresponding person and action label. We design a preliminary model and propose an improved model to infer 3D human poses in real time. When validating our method on the Invariant Top-View (ITOP) dataset, we found that compared with the initial model, our improved model improves the $mAP@10cm$ by 29%. When testing on the Kivi dataset, our improved model improves the $mAP@10cm$ by 15.74% compared to the preliminary

model. Our improved model can reach 65.89 frames per second (FPS) on the TensorRT platform.

# Acknowledgements

I would like to express my sincerest appreciation to my supervisor, Professor Abdulmotaleb El Saddik, for his guidance, support and patience for my whole graduate study period and in my personal life. His broad scientific vision and rigorous attitude have deeply affected me. He gave me many opportunities to participate in many projects, which greatly improved my academic and engineering skills. I am very proud of being one of his students.

I would like to thank all the colleagues I worked with who have left or are still working in the MCR Lab for their assistance. Specially, I would like to thank Dr. Haiwei Dong, who gave me a great deal of guidance when I entered a new academic field with difficulty. The suggestions he gave me will benefit me for life.

Finally, I would like to thank my family. Their love and understanding is my motivation for learning. I would not have made my achievements without them.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

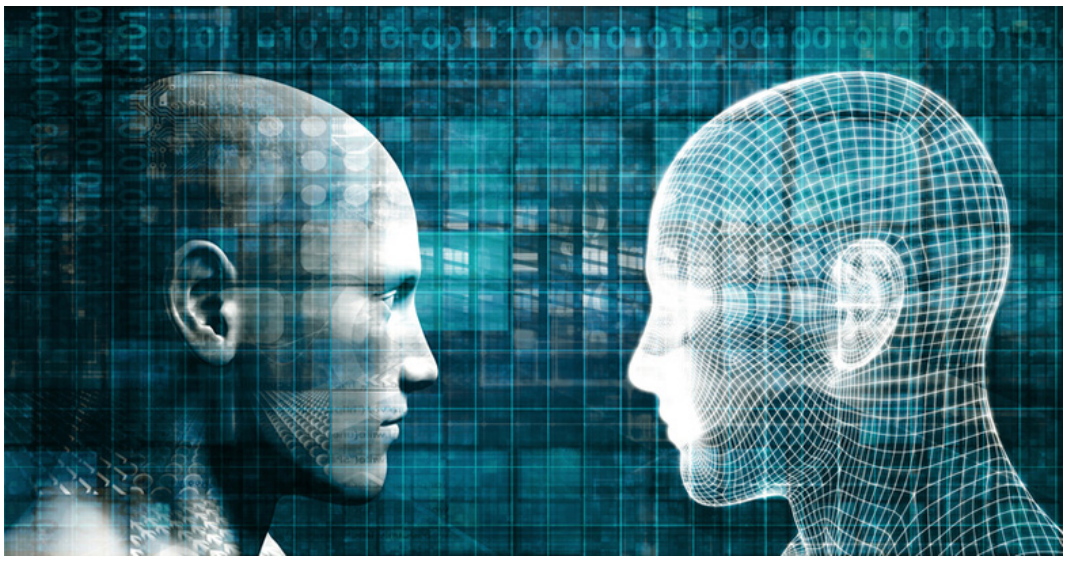
# Introduction

This chapter gives an overview of this thesis. In Section 1.1, we introduce the background and motivation of our work. Then, we report our contributions in Section 1.2. We describe the thesis outline in Section 1.4.

## 1.1 Background and Motivation

Digital twins (fig 1.1) were first introduced in the early 2000s for the manufacturing industry [8]. The concept was significantly expanded by Prof. El Saddik in 2018 for

(a) [41]



(b) [39]

Figure 1.1: Digital twins of industrial products (a) and humans (b)

digital smart cities [6]. The central change was that the new concept bridged living physical entities and the virtual world, which goes beyond manufacturing. An important example is digital twins of humans. The information and data obtained from humans can be applied to simulate and predict the health conditions of real humans.

The key step of digital twin technology is to extract analog information as digital

data for storage and analysis. The creation of digital twins involves research on human behavior and recognition. As early as the 1970s, Johansson [18] put devices with bright spots on the joint points of interest on the human body and then studied the behavior of the human body by observing the movement of the bright spots. Because this method of attaching special marked points to the human body is not convenient for application, researchers have turned their attention to research on non-contact recognition technology. Prior to the use of depth cameras, vision-based human pose recognition methods mainly used color image processing to obtain useful data features and use them as characteristic features. The content representation of an image usually involves some classic algorithms, such as local binary patterns (LBPs) [35], histograms of oriented gradient (HOGs) [5], Haar-like features [47], scale-invariant feature transforms (SIFTs) [26] and speed up robust features (SUFT) [1]. With the advent of depth cameras and Kinect [50] in particular, researchers have developed many new recognition technologies. For example, Raptis [37] used Kinect to obtain joint point coordinate information to calculate the key angle of the skeleton to recognize a human pose. The depth information obtained by a depth camera is an important reference dimension for human pose estimation.

Not only has the development of hardware brought changes to human body gesture recognition, but the maturity of deep learning has also made this technology develop faster. Based on deep learning technology in the computer vision community, the high-level digital data of humans—such as age, emotional state, appearance, and pose—can be estimated easily, even from a single image. This means that assessing the physical state, mental health, aesthetic expression, and behavioral patterns of a human will become increasingly direct and fast. The research on human behavioral patterns, including recovering the key points of a human from a given image, called pose estimation, was begun by Rosales and Sclaroff [38], who mapped low-level visual features to a high-level representation such as a set of joint positions of the body. A pose estimated with high confidence could be expanded to apply to more advanced areas, such as action recognition, pedestrian path detection, clothing parsing, games and animation, and human-computer interaction. With this background, AI-derived pose estimation could be helpful in physical education and health recovery. In traditional physical education and health

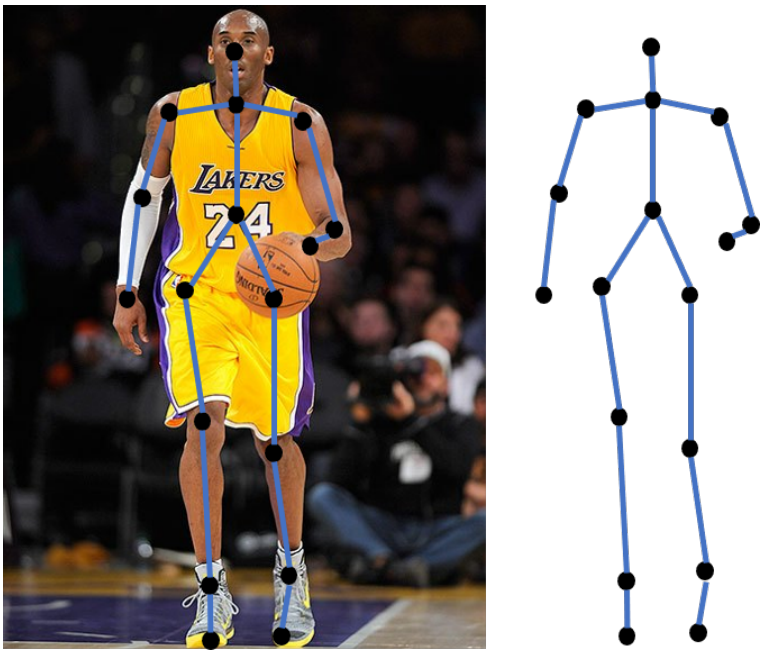



Figure 1.2: 2D pose estimation example. The pose is represented by keypoints. We can easily determine which joint is in the front and which joint is in the back because of the perspective view of the photo. However, when we remove the background photo, the 2D estimated pose shows only the 2D joint relations.

recovery, a normative action is subjectively corrected by the coach or the rehabilitation therapist. Pose estimation would greatly reduce labor costs and subjective judgment errors. 2D pose estimation technology allows computers to convert images to keypoints representing poses, as shown in Fig 1.2. Nevertheless, neither humans nor computers can recognize whether the joints are far or close due to missing perspective views. This is why researchers need to study 3D pose estimation. The 3D pose includes the depth axis information of joints so that it can eliminate ambiguity in the understanding of the pose. However, modern computer vision technology for 3D pose estimation in physical education or health recovery is still far from achieving human performance levels due to deficient datasets and inefficient models.

Popular datasets for single-person 3D pose estimation include Human3.6m (2014) [16] and ITOP (2016) [9]. The effectiveness of deep learning depends on the diversity of the dataset and the sufficient number of samples. Human3.6 m contains 15 daily actions,

Figure 1.3: Fast motions that cause optical blur [46]

and ITOP contains 15 small-scale actions. These datasets include color data collected by a monocular camera. However, fast movements can cause these pictures to have optical blur problems (fig 1.3), and these datasets do not contain fast actions. To fill this gap, it is necessary to build a new dataset containing fast move actions for physical education.

For our proposed dataset, the network can only learn the position of joints from a depth image because the dataset does not have two-dimensional color pictures. Because there is no light or color information, the segmentation of each part can only depend on the distance from the part to the camera in the depth map. Due to the inevitable measurement noise, the depth map has less effective information than a color picture; thus, it is a challenging task to estimate the pose of the human body from only the depth map. The low information content and high noise of depth images make it difficult for a simple network structure to effectively learn the target. Therefore, we design a hybrid model to estimate a 3D human pose from a depth image.

## 1.2 Contributions

The contributions of this thesis are as follows:

- We propose a new dataset for training and testing neural network layers for 3D human pose estimation. The dataset involves fast motions, which have a wider range of limb angles than slow motions. The depth image is acquired by a TOF camera, and the three-dimensional coordinates are acquired by a motion capture system. All of the images and labels have been synchronized.

- We propose the pseudo-3D soft-argmax function to avoid adapting 3D heat maps and to reduce the computational cost in the task of 3D human pose estimation.

- We design and train a convolutional neural network that can estimate 3D poses from a depth image. The model combines the benefits of a heat map-based model and an end-to-end model that can be deployed easily and has high-precision results. The model is verified to run in real time with acceleration by the TensorRT framework.

- We design and implement a user interface to show the estimated pose offline. The user interface displays a visualized depth map, a 3D human pose map that can be used to rotate the viewing angle and specify the three marginal heat maps of the joint, and a list of the estimated and ground-truth coordinates of the joints.

This thesis also includes collaborative work on Kivi dataset generation by me, Ph.D. student Matthew Mavor, Ph.D. student Gwyneth Ross, and Master's student Yufan Zhou under the supervision of Dr Abdulmotaleb El Saddik and Dr Ryan Graham. In this collaboration, I have mainly focused on software processing over data collection and data alignment, while Matthew Mavor and Gwyneth Ross spent countless hours collecting and cleaning the data from the Vicon motion capture system. Yufan Zhou collected and analyzed the data from Kinect.

## 1.3 Thesis Statement

3D human pose estimation from color images involves optical blur for fast motions; however, existing datasets contain only everyday slow actions. We aim to build a dataset comprising fast motions and show that a model that relies only on depth images can implicitly infer the 3D space relations and estimate the 3D human pose and that this model can run in real time.

## 1.4 Thesis Outline

The thesis is organized as follows:

- Chapter 2 describes the related work of 3D human pose estimation technology, including popular datasets and state-of-the-art methods applied to estimation from depth images.
- Chapter 3 presents the proposed dataset, which is called the Kivi dataset, including the collection procedure and the data cleaning details.
- Chapter 4 introduces two hybrid models we designed for 3D human pose estimation from depth images.
- Chapter 5 provides experiments and results for our models on two datasets. The results show the effectiveness and efficiency of our models.
- Chapter 6 draws conclusions and discusses the limitations of the proposed work and describes future work for improvement.

# Chapter 2

# Related Work

The challenging task of human pose estimation has essential theoretical value and a wide range of application scenarios. We can divide the current research on human pose estimation into two categories based on the final estimated human pose dimensions. These are 2D human pose estimation and 3D human pose estimation. 3D human pose estimation is more accurate for human expression than 2D human pose estimation, so its research and application value is higher than that of 2D human pose estimation. Researchers have used traditional computer vision algorithms for human pose estimation tasks and achieved results. In recent years, the use of deep learning methods for 3D

human pose estimation has also made significant progress. However, because the current 3D human pose labeling dataset has certain limitations compared to the 2D human pose labeling dataset, the difficulty of 3D human pose estimation is also greater than that of 2D pose estimation. Therefore, most current 3D human pose estimation methods are based on the more sophisticated 2D human pose estimation method. At the same time, whether it is 2D human pose estimation or 3D human pose estimation, it is necessary to segment and perform feature extraction on the human body position in a picture.

In this chapter, we first give an overview of human pose estimation, including the definition of human pose estimation and the difficulties encountered in the current research on human pose estimation. Then, we introduce human body segmentation methods and feature extraction methods, as well as deep learning research methods. Then, we introduce the common datasets and different methods used from the perspective of 2D and 3D human pose estimation. Finally, we introduce the model-related technologies used in this thesis, including the hourglass structure and marginal heat map.

## 2.1 Overview of Human Pose Estimation

The task of human pose estimation is to process a static image (including RGB images, depth images, etc.) or video containing human body information and then use a specific body posture estimation method to estimate the joints of the human body and connect adjacent joints according to a particular arrangement to form a complete body posture. If the position information of these nodes is only two-dimensional position information on the plane, it is called two-dimensional body posture estimation. If, in addition to two-dimensional position information, the point position also contains depth information, it is called three-dimensional human posture estimation. Therefore, according to the difference in the final estimated human posture dimension, the current research on human posture estimation can be divided into two categories: 2D human posture estimation and 3D human posture estimation.

Human pose estimation is different from human pose or motion recognition. The estimation only aims to estimate a complete human skeleton from the original data.

Abstractly, it is a regression problem, but recognition is performed by using one or more data sources, such as the original images and videos, or by using human skeleton information directly; a mix of various information can even be used to understand human poses, and this can be abstracted into a classification problem. If understanding of multiframe poses is added based on a single frame, ordered gestures can be combined into one action, so the recognition of ordered gestures in multiple frames can be considered motion recognition. Among all the information that can be used in human pose or motion recognition, human pose information can uniquely represent the current state of the human body, and its accuracy dramatically affects the accuracy of recognition. Therefore, human pose estimation can be regarded as one of the essential and basic tasks of human pose or motion recognition.

At present, a large number of researchers are researching human pose estimation. Many innovative methods have emerged in both 2D and 3D human pose estimation[48][2]. However, these studies are currently being applied in actual production. There is still a certain degree of difficulty in life. Some of these difficulties are caused by the characteristics of human pose estimation itself. On the other hand, they are also caused by incomplete research conditions. The specific difficulties are as follows:

- Complex background interference. This is an unavoidable problem in the application of human pose estimation and actual scenes. When people are the target of research, the color of their clothes is unpredictable, and at the same time, the background color is unpredictable. In some cases, the color of the clothes is very similar to the background color. It is difficult for some human pose estimation methods to effectively separate the human body from the background, which leads to a certain degree of deviation in the final human body pose estimation results.

- Occlusion issues. We divide occlusion into two types. One is the self-occlusion problem caused by the structure of the human body. For an object with a complex structure such as the human body, if it is observed from one perspective, there may be a certain degree of occlusion. For example, when observing the human body from the back, the human hand may be blocked by the trunk. Another is the prob-

lem of human occlusion caused by objects. For example, when the human body is behind a table, the table may occlude the human limbs and part of the torso. There are also different degrees of occlusion problems. For example, half-body occlusion can be considered severe occlusion. When encountering serious occlusion problems, some human pose estimation methods use the strategy of directly discarding the severely occluded body parts. When the occlusion is not as severe, the position of the occluded part cannot be inferred from the original features of the image. The current method uses the inherent correlation information of the joint points adjacent to it as a priori information to obtain an approximation of the location.

- Diversity of appearance. The appearance is influenced by certain factors when performing human pose estimation. The first factor is the effect of lighting. Since the lighting environment is often very complicated in daily environments, there may be effects such as reflections, shadows, and low light, which will affect the accurate estimation of the position of human joint points. Second, there are differences in the clothing of different people, and these differences will be reflected in aspects such as the color, shape, and material of the clothes, which will also affect the estimation results. Finally, there are differences in the sizes of people. If a single dataset is used for training, the trained network model may have a certain degree of error when estimating different human bodies.

- Limitations of perspective. Due to the limitations of the current dataset, the original image data used for human pose estimation are collected from some conventional perspectives. However, in actual application scenarios, the perspective is often not fixed, and there may be some unusual angles. The generalization ability of the network model plays a vital role in accurately estimating the pose of the human body when an unusual perspective appears.

- Dataset limitations. This problem is easy to solve for 2D human pose estimation, in which it is easy to generate labeled information. At present, there are also a large number of datasets used for training in the 2D human pose estimation task.

A large number of these datasets face the above four problems; consequently, the above problems can be solved by improving the network structure and the feature extraction method. However, for 3D human pose estimation, the scale of the labeled datasets is far from sufficient. Because an accurate 3D human pose needs to be collected in a laboratory environment as the ground-truth value of training, the dataset lacks the diversity of the above four points to a certain extent.

Therefore, when researching 3D human pose estimation methods, the question of how to use a limited dataset to train a 3D human pose estimation model with high accuracy and a good generalization ability is a popular and challenging problem in current research.

## 2.2 Deep Learning Method

In recent years, research on deep learning has strongly promoted the practical application of computer vision in various fields. The earliest neural networks, which are the core of deep learning, can be traced back to the 1940s. The psychologist Donald Hebb proposed Hebbian learning. Then, researchers created the perceptron based on Hebbian learning. However, because the computers at the time did not have sufficient capacity to handle the calculations required for the construction of multilayer neural networks, the development of neural networks was slow before the emergence of stronger computers. It was not until 1986 that Rumelhart and Hinton et al [40] applied the backpropagation algorithm to artificial neural networks for the first time, solving the problem that a two-layer neural network required complex calculations, so that the research on artificial neural networks could continue. By 2006, the “deep belief network" proposed by Hinton was different from the traditional neural network used in training [12]. The deep belief network has a pretraining process and a fine-tuning process to optimize the training of the entire network, significantly reducing the training time. Deep confidence networks gave a new name to the learning method of multilayer neural networks, “deep learning." By 2012, Hinton’s team used AlexNet [22], a deep learning network built from convolutional neural networks (CNNs). AlexNet, for the first time in the ImageNet competition,

trained one million pictures containing one thousand categories, far exceeding the second-place system (using the support vector machine method), which sufficiently proved the superiority of deep learning in the field of image recognition.

After the development of deep neural networks in recent years, many different network types have been derived, such as CNNs, commonly used in the field of image recognition; recurrent neural networks, commonly used in the field of natural language processing; and the length of time, commonly used in the fields of speech recognition and video understanding. Memory networks, generative adversarial networks for picture and video generation, and others have also been used. As the most widely used convolutional neural network in the field of computer vision, the earliest modern CNN model structure was LeNet, which was proposed by Lecun in 1998 [23]. The convolutional layer, pooling layer, activation layer, fully connected layer, and loss function layer in the paper laid the foundation for the use of deep learning in the field of computer vision.

Convolutional layers: Each convolutional layer in a CNN consists of several convolutional units. The input of each layer is calculated by a convolution kernel on the upper-level input layer by sliding windows that are adjusted one by one. In the convolution kernel, each parameter is multiplied by the corresponding local pixel value to obtain the result for the convolution layer. The optimization of the backpropagation algorithm yields the parameters of each convolution unit. The purpose of the convolution operation is to extract the different features of the input. The first convolution layer may only extract some low-level features such as edges, lines, and corners. Adding layers to the network can enable the iterative extraction of more complex features from low-level features.

Pooling layer: Pooling is another crucial concept in CNNs. It is a form of down-sampling. After obtaining the features of the image through the convolutional layer, in theory, we can use these features to train the classifier directly. This may make the trained model overfitted. To further reduce the network training parameters and the degree of overfitting of the model, the convolutional layer needs to be pooled. In the convolutional neural network, a pooling layer is generally inserted periodically to continuously reduce the magnitude of the data. At the same time, the number of parameters

and the amount of calculation in model training will also decrease accordingly. There are usually two methods of pooling: maximum pooling divides the input image into several rectangular regions and outputs the maximum value for each subregion. Average pooling outputs the average value to the divided rectangular area. In practical applications, because the effect of maxpooling is better than that of average pooling, most of the current convolutional neural networks use maxpooling.

Activation layer: The core of the activation layer is the activation function. The role of introducing the activation function is to add nonlinear factors to the neural network, because the convolution layer of a CNN essentially performs multiplication and addition on different pixels. These operations are linear, and in practical problems, the data are generally not linearly separable. To address this problem, we can add an activation layer to the neural network to introduce nonlinear factors. The activation function should have certain properties such as being nonlinear, continuously differentiable, monotonic, and approximately linear at the origin point as well as having an unsaturated range. Currently, commonly used activation functions are sigmoid, tanh, ReLU, and leaky ReLU.

Loss function layer: The loss function layer punishes the network parameters by measuring the difference between the network prediction result and the real value during the training process. The update of the overall network parameters is aimed at minimizing the loss function, so the loss function layer is often the last layer of the network. The design of the loss function has a decisive influence on the network training results. Different loss functions are suitable for different tasks. For example, the softmax cross-entropy loss function is often used for the task of selecting one of K categories, and the sigmoid cross-entropy loss function is often used for multiple independent binary classification problems. The mean-square error function is often used for estimating the actual value. For complex problems, different loss functions may need to be mixed to achieve the best results.

The current research on deep learning mainly focuses on the following areas: 1) strengthening the network structure in depth or breadth so that it can capture more information from the original data, 2) effectively deepening the network while effectively solving the gradient vanishing problem in the process of backpropagation, 3) optimizing

the storage performance of the pretrained network model so that the model can be used on devices with weak storage and computing capabilities, and 4) applying deep learning in certain specific fields, which requires the network structure and loss function to be modified to adapt to these specific scenarios. After 2012, researchers proposed many excellent influential network structures, for example, 1) VGGNet [43]: this network structure is very mobile and has excellent generalization capabilities; 2) GoogLeNet [45]: the proposed network, through NIN [25] and stacking the convolution and pooling operations that are commonly used in CNNs, on the one hand, increases the width of the network and on the other hand, increases the adaptability of the network to different scales; 3) ResNet [10]: this is mainly used for deep layers and addresses the problem of network degradation that occurs during network training. A residual network improves network performance by introducing a short-circuit layer based on the VGG19 network. In addition to proposing deeper and wider networks, there are also studies dedicated to improving network inference efficiency and storage performance, such as Xception [3] and MobileNet [13].

## 2.3 3D Pose Estimation

3D human pose estimation is a subset of human pose estimation. It uses a 3D skeleton to describe the human pose, which is more accurate than a 2D human pose. However, due to occlusion and missing datasets, the human pose must be estimated. 3D estimation is much more complicated than 2D human pose estimation. 3D human pose estimation in deep learning has gradually become the mainstream method. At present, research on 3D human pose estimation from depth images using deep learning has mainly focused on two directions. One is end-to-end 3D human pose estimation. The second is staged 3D human pose estimation. The following introduces a 3D human pose annotation dataset and 3D human pose estimation based on deep learning.

### 2.3.1 3D Human Pose Annotation Dataset

In the data acquisition and processing phase, 3D human pose estimation is more complicated than 2D human pose estimation. Many of the datasets used for 2D human pose estimation are collected in a natural environment, and obtaining their labels is simple. Due to the complexity of 3D human body annotations, almost all of them are collected in a laboratory environment. The acquisition process requires a large number of sensors and cameras, which ultimately results in a 3D human body attitude annotation dataset; both diversity and quantity are much lower than those of 2D datasets. Currently, common 3D human pose annotation datasets include the CMU Panoptic dataset [20], MPI-INF-3DHP [28], HumanEva [42], Human3.6m [16], and ITOP [9].

- CMU Panoptic dataset. CMU produced this dataset. Each collected scene is a spherical space, with a total of 480 VGA cameras, 31 HD cameras, and 10 Kinect II sensors for collection. The dataset contains a total of approximately 5.5 hours of video data, of which there are approximately 1.5 million different 3D human skeletons. In addition to 3D human pose data, the dataset also provides 3D facial data for face recognition and hand joint point data for gesture tracking and recognition.

- MPI-INF-3DHP. This dataset was produced by the Max Planck Institute for Informatics. A professional motion capture system collected it. Unlike the general dataset, which contains both indoor and outdoor scenes, the test set includes a total of 2929 frames. These frames contain six different characters, with a total of 7 behaviors.

- HumanEva. This dataset has two parts, HumanEva-I and HumanEva-II. HumanEva-I contains seven different verified video sequences, and its 3D human pose marker data are collected through a motion capture system. This dataset contains a total of 4 different characters and a total of six daily actions. This dataset contains a test set, validation set, and training set, while HumanEva-II contains only a test set.

- Human3.6m. Human3.6m is currently the largest 3D human pose labeling dataset. This dataset contains approximately 3.6 million 3D human poses and their corresponding image data, with a total of 11 different actors. Each actor in the dataset has 15 action scenes. The images in the dataset were taken by four digital cameras fixed at the four corners of the venue and synchronized by a time sensor. The 3D human body pose annotation data were collected by a motion capture system containing ten motion cameras.

- ITOP. This dataset consists of 50K depth images for training and 10K depth images for testing in the front/side view and top view. Both views recorded 20 people performing 15 different actions with two Asus Xtion Pro cameras. As the ground truth, the 3D coordinates of 15 joints are labeled in real-world meters.

### 2.3.2 3D Human Pose Estimation Based on Deep Learning

The core of deep learning is deep neural networks. The structure of neural networks has been proven to obtain deep-level features from images effectively, and complex tasks such as 3D human pose estimation are sensitive to structural features. It is necessary to improve the network structure to capture features in a more extensive range. On the other hand, it is also possible to introduce geometric constraint information into a network by transforming the loss function to achieve more accurate 3D human pose estimation. Therefore, there are two core problems of 3D human pose estimation based on deep learning: one is the design of the network structure, and the other is the design of the loss function.

The current 3D human pose estimation task can be divided into three types by the input type. One takes color images as the input. Since the input has only 2D information and the output is three-dimensional, this type of task is an underconstrained problem. An estimated 3D human pose often has higher estimation accuracy on the X and Y axes. The accuracy on the Z-axis, representing depth, is not high. To solve this problem, researchers have introduced depth images to assist in training networks to increase the network's estimation accuracy for three-dimensional coordinates. Both of the above tasks

estimate the 3D human pose from a 2D color image. Some researchers believe that with the continuous development of depth cameras, estimating 3D human poses from depth images is also of enormous research value. The third type of 3D human pose estimation takes only depth images as input.

The current research on 3D pose estimation based on deep learning can be divided into three categories according to the training method of the 3D human pose network model: end-to-end 3D human pose estimation, heat map-based 3D human pose estimation, and hybrid model-based 3D human pose estimation.

#### 2.3.2.1 End-to-end 3D Human Pose Estimation

An end-to-end network is similar to a black box. The input of the network is the original image data, and the output is the output data expected by the task. For the 3D human pose estimation task, the output should be the 3D position coordinates of all corresponding human joint points. An end-to-end network uses a vast network structure to process all data content, and there are no other intermediate data used in processing. Therefore, for such methods, the design of the network structure and data preprocessing are crucial. At present, end-to-end 3D human pose estimation network structures are mainly based on the regression network structure, and the 3D positions of the joint points are estimated directly from the image regression [24][4][44].

#### 2.3.2.2 Heat Map-based 3D Human Pose Estimation

In human pose estimation, the heat map shows the probability of a joint occurring at each spatial location in the image. The input of the network is a depth map, and the output of the network is a heat map. Compared to directly regressing the points' coordinates, the process of generating a heat map is performed with only a convolution operation. Hence, the network converges faster than with the regression method. Heat maps map the original full image. There may be correlations between the joint points in the heat map, and these correlations can be used to guide network training implicitly. Heat maps also capture the relationship between the foreground and background and can be used to guide network training. However, the feature map inferred by this method cannot

establish a direct relationship with the coordinate values of keypoints. To obtain the position of the most likely point from the heat map, $argmax$ is typically used, and this operation is not derivative. Therefore, this kind of network is not end to end [31].

#### 2.3.2.3 Hybrid Model-based 3D Human Pose Estimation

In 2D human pose estimation, to make use of the information obtained by the upper heat map in the network and to enable end-to-end pose estimation, researchers have developed a hybrid model [27]. The hybrid model uses soft-argmax to process the 2D heat map generated by the network and obtain numerical coordinates. Because soft-argmax is differentiable, it can be integrated into the above network for generating heat maps and end-to-end estimates of human coordinates. This technology has been successfully applied to 3D human pose estimation with color pictures as input [33].

## 2.4 Hourglass Structure

The hourglass structure was proposed for solving the 2D pose estimation problem by Newell et al. [32]. The shape of the structure is similar to that of a horizontal hourglass. The module was designed for capturing different scales of information. To predict the posture of an entire person, the network must not only be able to obtain local information, such as the position of the hand, but also hidden clues, such as the distance between joints, the orientation of the human body, the arrangement of the limbs, and the angle of the joints. The hourglass structure can extract local features from images and provide a coherent understanding of the whole body for pixelwise prediction in heat maps.

The whole processing system includes downsampling and upsampling. Several layers of convolution and maxpooling reduce large-scale data to a low resolution. After reaching the lowest resolution, the network uses nearest-neighbor upsampling to obtain high-resolution features from the lower-resolution features, followed by an elementwise addition operation that combines the high-resolution features and the features that have the same resolution when downsampling. An illustration of the hourglass module is

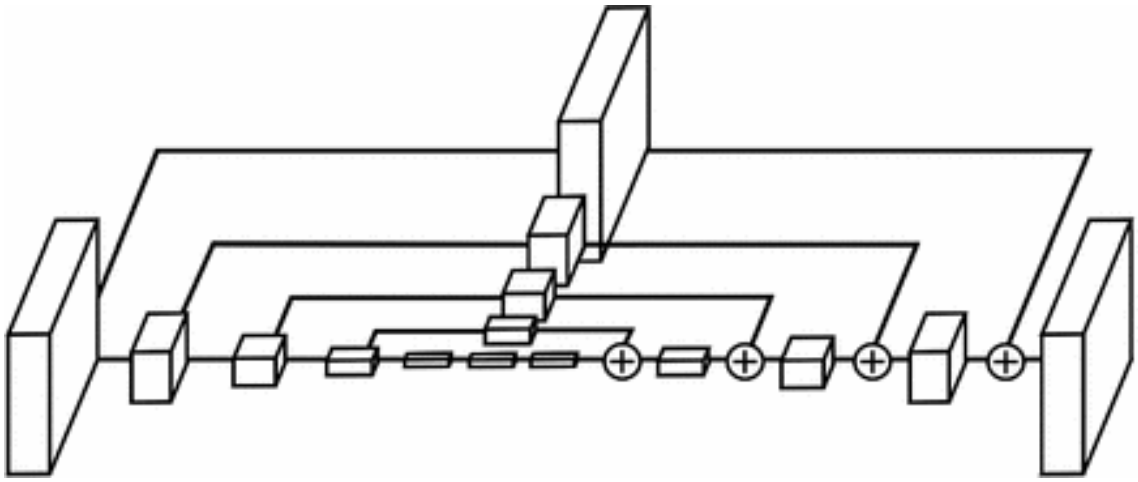


Figure 2.1: Illustration of the hourglass module. Each box in the figure represents a residual module. [32]

shown in Fig. 2.1.

## 2.5 Soft-argmax

The argument of the maxima, $argmax$, in the heat map is the index of the maxima that indicate the target point's coordinates. The $argmax$ operation traverses the matrix to obtain the position of the maximum value. Soft-argmax was proposed in [27] to obtain the maximum position through differentiable operations so that it can be integrated into the end-to-end network.

Consider normalizing the heat map $h \in \mathbb{R}^{WxH}$ as a probability mass function $P$ with the $Softmax$ operation:

$$P(h_{i,j}) = \frac{e^h_{i,j}}{\sum_{k=1}^{W}\sum_{l=1}^{H} e^{h_{k,l}}}. \tag{2.1}$$

If the index is treated as a random variable, its mathematical expectation can be calculated by soft-argmax as follows:

$$\mu_x = \mathbb{E}[X] =< P \cdot T_x >, \mu_y = \mathbb{E}[Y] =< P \cdot T_y >, \tag{2.2}$$

where $< \cdot >$ is the scalar product. $T_x, T_y$ are the coordinate indicator tensors, defined as follows:

$$T_x(i,j) = i/W, T_y(i,j) = j/H. \tag{2.3}$$

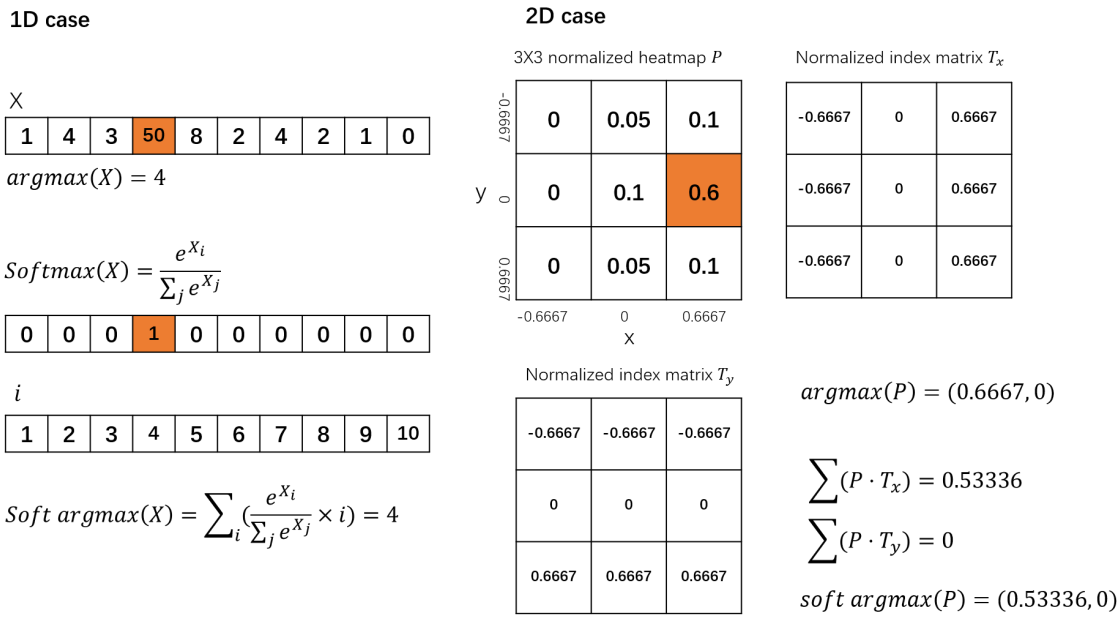



Figure 2.2: Graphical depiction of soft-argmax calculations on 1D and 2D heat maps.

$Soft - argmax$ can be integrated into the neural network by using backpropagation and the chain rule on the derivative with respect to $h$:

$$\frac{\vartheta \mu_x}{h_{i,j}} = T_x \frac{e^{h_{i,j}} (\sum_{k=1}^{W} \sum_{l=1}^{H} e^{h_{k,l}} - e^{h_{i,j}})}{(\sum_{k=1}^{W} \sum_{l=1}^{H} e^{h_{k,l}})^2} \tag{2.4}$$

$$\frac{\vartheta \mu_y}{h_{i,j}} = T_y \frac{e^{h_{i,j}} (\sum_{k=1}^{W} \sum_{l=1}^{H} e^{h_{k,l}} - e^{h_{i,j}})}{(\sum_{k=1}^{W} \sum_{l=1}^{H} e^{h_{k,l}})^2}. \tag{2.5}$$

With the integration of soft-argmax, the neural network can estimate coordinates directly without external calculations.

To represent a 3D pose with a 2D heat map, Aiden et al. [33] considered three heat maps to represent a 3D pose voxel in three views, as shown in Fig. 2.3. The three heat maps are the projections from the directions of the normals of the three faces, and they are

$$\begin{aligned} \hat{H}^{(xy)} &= \sum_i \hat{H}_{i,:,:} \\ \hat{H}^{(zy)} &= (\sum_k \hat{H}_{:,:,k})^T \\ \hat{H}^{(xz)} &= \sum_j \hat{H}_{:,j,:}. \end{aligned} \tag{2.6}$$

Corresponding to the soft-argmax calculation method of the 2D heat map, two coordinates can be calculated for each volumetric heat map. The heat map is normalized to sum to one at the beginning. For example, $\hat{H}^{(xy)}$ can yield the x-coordinate and y-coordinate. To estimate 3D coordinates from a color image, the x-coordinates and y-coordinates are computed from $\hat{H}^{(xy)}$ due to having the same orientation, whereas the z-coordinates are the averages of the other heat maps:

$$\begin{aligned} \mu_x &= < \hat{H}^{(xy)} \cdot T_x > \\ \mu_y &= < \hat{H}^{(xy)} \cdot T_y > \\ \mu_z &= \tfrac{1}{2}(< \hat{H}^{(zy)} \cdot T_z > + < \hat{H}^{(xz)} \cdot T_z >). \end{aligned} \tag{2.7}$$

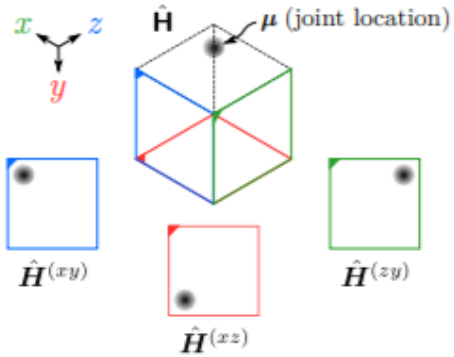



Figure 2.3: Volumetric heat maps can fully represent a 3D Gaussian in a voxel [33]

To estimate a 3D pose from a depth image, we slightly improved this method. We will introduce the improvement in section 4.2.2.

# Chapter 3

# Kivi Dataset

In this chapter, we propose a dataset for human pose estimation. In Section 3.1, we give an overview of the human kinetic dataset, Kivi, which is used to effectively evaluate the performance of different neural network models for the task of human pose estimation. We then introduce the collection environment and the procedure of collection. Then, we introduce data labeling and data synchronization in Section 3.3.

## 3.1 Introduction

We thank Ph.D. student Gwyneth Ross and Ph.D. student Matthew Mavor from the School of Human Kinetics and Master's student Yufan Zhou from the School of Engineering for collaborating in collecting the data. The proposed dataset costs 29.1 GB of space and includes 66839 frames; 12 people performed 38 actions. We divided the dataset into a training set and test set by subject. The training set consists of 10 subjects, including 52562 frames, and the testing set consists of 2 subjects, including 14277 frames.

The details of the proposed dataset are shown in Tables 3.1, 3.2, and 3.3.

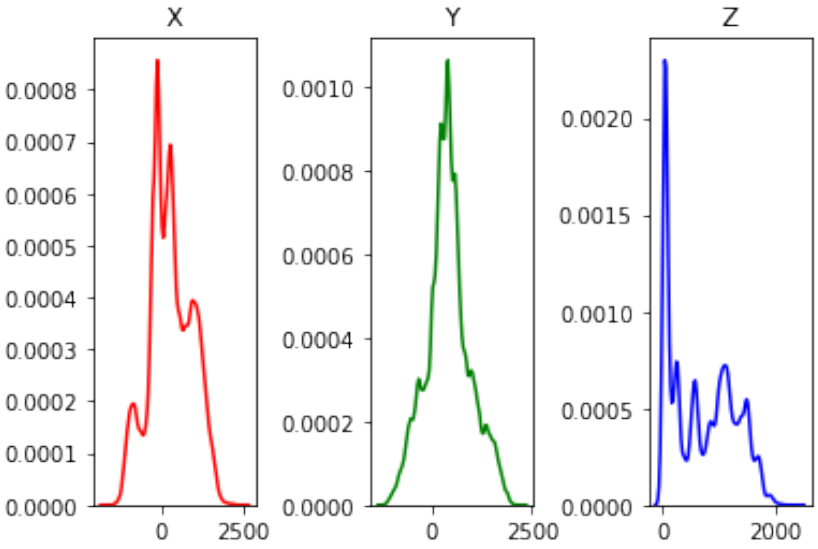
Table 3.1: The file size of the proposed dataset

| Split | Frames | Subjects | Actions | Depth Maps | Labels |
|---|---|---|---|---|---|
| Train | 52562 | 10 | 38 | H5 (22.8 GB) | H5 (24.0 MB) |
| Test | 14277 | 2 | 38 | H5 (6.2 GB) | H5 (6.5 MB) |

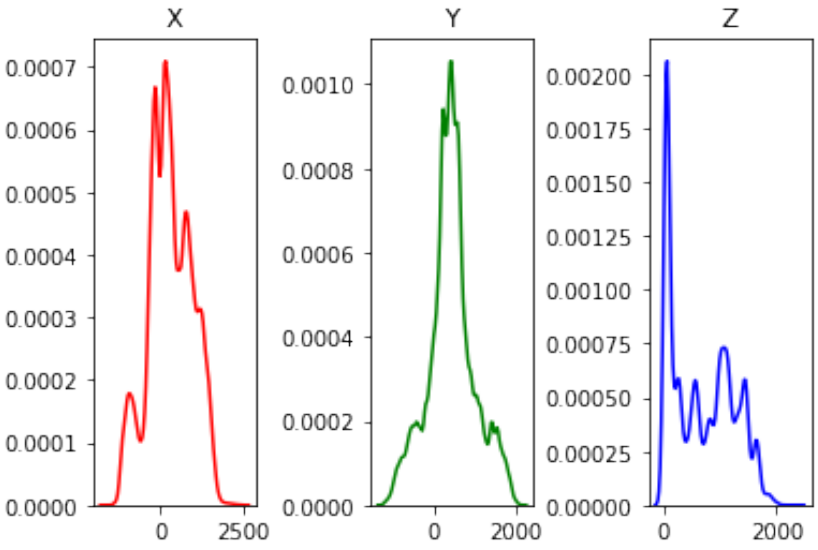
Table 3.2: Dimensions, attributes, and data types in the depth map H5 file

| Key | Dimensions | Data Type | Description |
|---|---|---|---|
| id | (n,) | uint8 | Frame identifier of the form SSMMXXXXX, where SS is the person's ID number, MM is the motion ID number, and XXXXX is the frame number. |
| data | (n, 424, 512) | float16 | Depth map corresponding to a single frame. The depth values are in real-world meters (m). |

To see if there is a significant difference between the training and testing sets, we use kernel density estimation, a nonparametric test method, to infer the data distribution patterns of the training and testing sets. We divide the coordinate labels by dimension, tile them into a one-dimensional vector, and derive their estimated kernel density distributions, as shown in Fig 3.1.




(a) Estimated kernel density distribution of the training set




(b) Estimated kernel density distribution of the testing set

Figure 3.1: Estimated kernel density distribution of our Kivi dataset.

Table 3.3: Dimensions, attributes, and data types in the label H5 file

| Key | Dimensions | Data Type | Description |
|---|---|---|---|
| id | (n,) | uint8 | Frame identifier of the form SSMMXXXXX, where SS is the person's ID number, MM is the motion ID number, and XXXXX is the frame number. |
| coordinates | (n, 18, 3) | float64 | Three-dimensional points (x, y, z) corresponding to the location of each joint in real-world millimeters (mm). |
| subject | (n, 1) | int8 | Person's ID number corresponding to a single frame. |
| motion | (n, ) | int8 | Motion ID number corresponding to a single frame. |

## 3.2 Data Collection

In this section, we describe the capture space and the recording conditions as well as the software design consideration.

The capture region and the placement of the MoCap and TOF cameras are shown in Fig 3.2. We captured the marker positions from 10 MoCap cameras and the depth data from 1 TOF camera. The laboratory area is approximately $8m \times 6m$, and within it, we obtained a region of approximately $4m \times 5m$ as the efficient and valid region according to the valid detected range of the TOF camera. The 10 MoCap cameras were suspended above the laboratory, 2 of them on each of the left and right sides, 2 of them on each of the front and back sides, and 2 at the middle top. The control station was at the back of the laboratory. The TOF camera was set up on a tripod that was 1.5 meters high and was set in front of the control station.

The reflective markers attached to the designated body parts of the subjects were used to record three-dimensional coordinates and trajectories with the MoCap cameras.

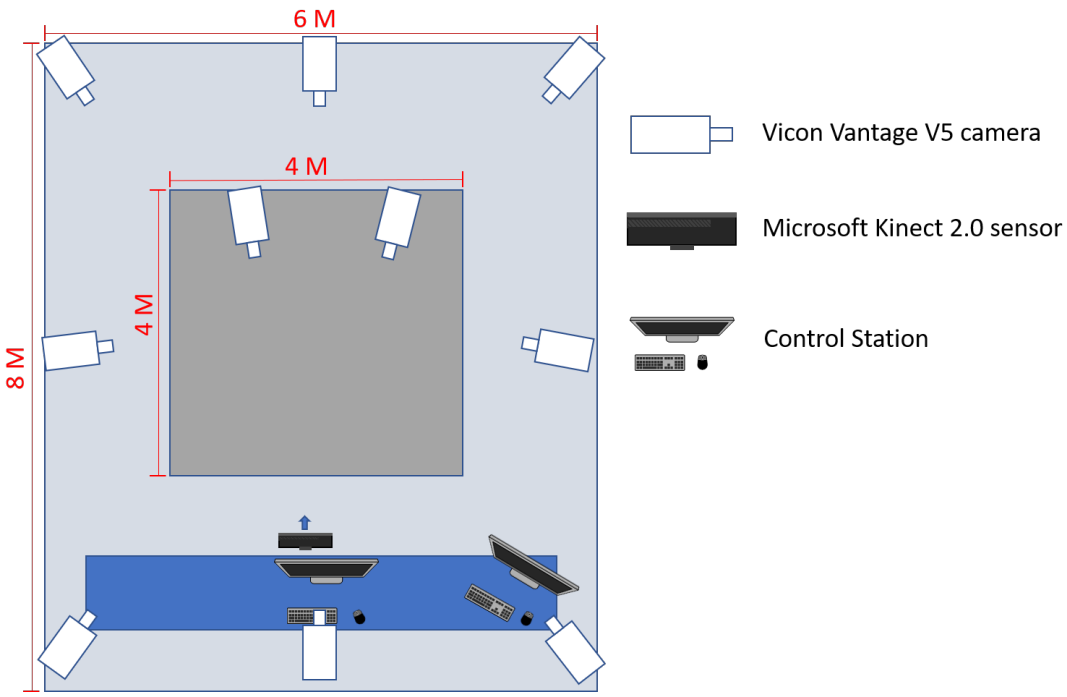



Figure 3.2: Floor plan showing the capture region and the placement of the TOF and MoCap cameras

We also put three markers on the TOF camera for computing the relative coordinates of the markers on the subjects. The markers were initially labeled manually and were automatically applied to all experiments.

We set up a server on a computer for retrieving data from MoCap cameras and a client on another computer for retrieving data from the TOF camera. We did not use the same computer for the two kinds of cameras because the TOF camera was not compatible with the operating system of the computer, which has the license for the MoCap cameras. The server and the client communicated with each other based on the TCP/IP protocol. The overall system architecture was designed as shown in Fig. 3.3. Multiple components were implemented in the system to ensure the data were reliable and easy to process during cleaning:

1) Transmission component: The transmission component establishes and maintains the connection between the server and the client. The data and the control commands are packaged and exchanged through the transmission component in an individual thread.

As the basis of the communication function of the system, the transmission compo-

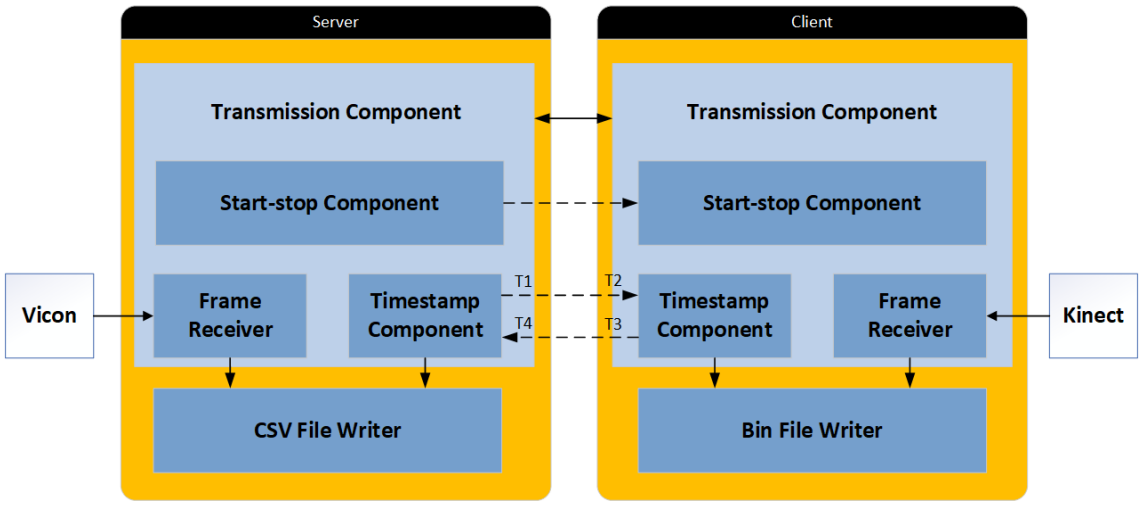



Figure 3.3: The overall system architecture of the Vicon-Kinect data collection system. The solid lines in the figure indicate the data transmission path. The dashed lines in the figure indicate indirect communication between the components, the data and the control commands communicated through the transmission component. The system was developed by Yufan and Jianquan.

nent is implemented based on the TCP/IP protocol and applies WinSocket APIs [19].

2) The timestamp component: The timestamp component counts the epoch time since 1970.1.1 of the local machine in microseconds (ms). The timestamp is not only used in recording the time of each frame but also plays a role in soft synchronization during data cleaning.

The timestamp is the basis for data alignment. After the server and client establish a connection, the first step is to obtain four timestamps to calculate the local time deviation between the client and the server. A timestamp attaches to each frame regardless of whether the data source is a moCap or TOF camera.

3) The frame receiver: Both the server and the client have a frame receiver in an individual thread. The server receives the marker location data from the moCap camera system. The client receives a depth map from the TOF camera. The data are combined with the timestamp and storage into a struct and then pushed into a queue to wait for retrieval by the file writer.

For retrieving data from the TOF camera, we apply Kinect SDK. For retrieving data

from the moCap cameras, we apply Vicon Data Stream SDK.

4) The CSV/bin file writer: The file writer retrieves data from the data buffer queue and writes the data into a CSV or bin file. The data from the moCap cameras are organized as frames × the markers' location matrix and are stored in a CSV file. The data from the TOF camera are stored in a binary file by frame.

5) The start-stop component: The server and client must start and stop at the same time, so the start-stop component is implemented in an individual thread controlled by the server.

The whole procedure of data collection is as follows: First, Gwyneth and Matthew calibrate the Vicon motion capture system and dress the participants in the Vicon suit and markers. Then, they digitize the markers in order to digitize the table. When those preparations are complete, the server issues the start command. After receiving the command, the client waits for the soft-synchronized data. After that, the synchronization mechanism computes the time offset between the server and the client. This time offset is recorded at the head of each CSV file. Then, the two systems begin to retrieve and write data until the server issues the stop command, and the two machines stop retrieving data at the same time. The file writer continues to retrieve all the data in the data cache and write it to a file for storage. Then, the system does not disconnect and waits for the next start command.

## 3.3 Data Labeling

Data labeling is composed of two steps. The first step is to transform the original markers into joints. The second step is to map the depth frame and the joints' coordinates via software synchronization.

### 3.3.1 Markers to Joints

Initially, the CSV files we obtain from the Vicon motion capture system contain the spatial coordinates of the markers and timestamps. However, what we need are the coordinates of the human body joints. We compute each joint's coordinates by computing

the average of some specific marker coordinates and generate new CSV files to store those joint coordinates. The correlation between the markers and joints is described below.

In total, we have 78 markers on the subject's body. However, not all of them appear on the same subject. For example, we have four markers named "RFHD", "LFHD", "RBHD", and "LBHD" attached to the subject's head. We also have four markers named "FHD", "RHD", "LHD", and "BHD" in another group. We use only four markers to represent the head, so only a group of marker names was chosen. We compute the location of the head by averaging the locations of the four markers. The same situation occurs for the other joints, and every joint has two groups of markers. The markers are used to calibrate and track body parts. The marker placement is shown in table 3.4.

As introduced above, the average of four markers gives the location of the head. Similarly, the torso's location is the average coordinates of two bones ($XP, T8$) or a cluster of markers ($LBUL, LBUR, LBLL, LBLR$). The shoulder's location is the average of the acromion ($LAC, RAC$) and the scapula ($LSCAP, RSCAP$). The hip's location is the coordinates of the iliac crest ($LIC, RIC$) or the average of ($PVUL, PVLL, PVUR, PVLR$). The ankle joint's coordinates are from the lateral malleolus ($LANL, RANL$) or the marker on the back part of the feet. We use a toe joint in the skeleton to indicate the frontal direction. The toe's coordinate is from the first metatarsal bone ($LM1, RM1$) or the marker on the front of the feet. The knee's location is the average coordinate of the lateral femur epicondyle ($LKNL, RKNL$) and the medial femur epicondyle ($LKNM, RKNM$) or the center point of the line between the markers on the thigh and the markers on the shank. The elbow's location is the average coordinate of the humerus lateral epicondyle ($LELBL, RELBL$) and the humerus medial epicondyle ($LELBM, RELBM$), or the center point of the line between the markers on the thigh and the markers on the shank.

After brute-force traversing all CSV files from the Vicon motion capture system and converting the markers to joints, we obtain CSV files with joint coordinates and timestamps. The joints are in the order of head, left foot, right foot, torso, left ankle, right ankle, left hip, right hip, left shoulder, right shoulder, left wrist, right wrist, left toe, right toe, left knee, right knee, left elbow, and right elbow. We label the depth

Table 3.4: Marker placement

| Joint | Placement | Name |
|---|---|---|
| Head | Front right/Front | RFHD/FHD |
| | Front left/Left | LFHD/LHD |
| | Back right/Right | RBHD/RHD |
| | Back left/Back | LBHD/BHD |
| Shoulder | Acromion | LAC/RAC |
| | Scapula | LSCAP/RSCAP |
| Elbow | Humerus Lateral Epicondyle | LELBL/RELBL |
| | Humerus Medial Epicondyle | LELBM/ RELBM |
| | Cluster: Middle, Lateral | LUAPP, LUAPA, LUADA, LUADP, LFAPP, LFAPA, LFADA, LFADP/ RUAPP, RUAPA, RUADA, RUADP, RFAPP, RFAPA, RFADA, RFADP |
| Wrist | Hand | LHAND/RHAND |
| Torso | Xiphoid | XP |
| | T8 | T8 |
| | Cluster:T10/T12 vertebrae | LBUL, LBUR, LBLL, LBLR |
| Hip | Iliac Crest | LIC/RIC |
| | Cluster: Sacrum | PVUL, PVLL/PVUR, PVLR |
| Knee | Lateral Femur Epicondyle | LKNL/RKNL |
| | Medial Femur Epicondyle | LKNM/RKNM |
| | Cluster: Middle; lateral side on thigh | LTHPA, LTHPP, LTHDA, LTHDP/RTHPA, RTHPP, RTHDA, RTHDP |
| | Cluster: Middle; lateral side on shank | LSHPA, LSHPP, LSHDA, LSHDP/RSHPA, RSHPP, RSHDA, RSHDP |
| Ankle | Lateral Malleolus | LANL/RANL |
| | Back part of foot | LFTB/RFTB |
| Toe | First metatarsal | LM1/RM1 |
| | Front part of foot | LFTF/RFTF |

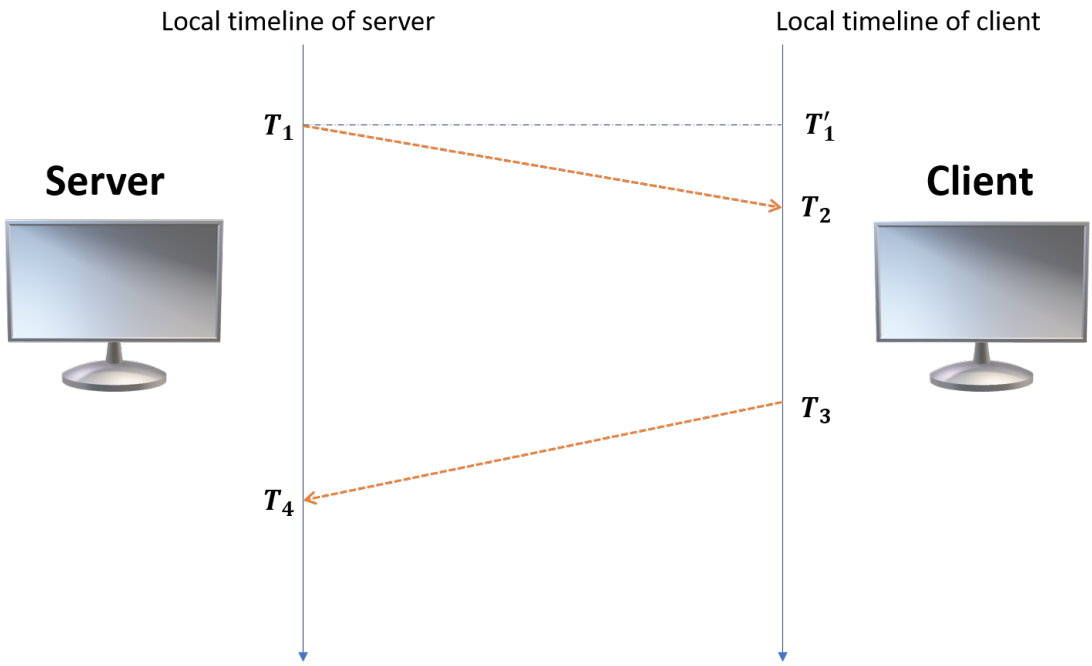



Figure 3.4: Synchronization mechanism of the customized scheme

frames with joint coordinates when performing software synchronization.

### 3.3.2 Software Synchronization

We need clocks for various social activities. In the field of test and measurement, high-precision timestamp signals represent the precise acquisition of physical processes and affect subsequent analysis and processing. It is necessary to define the clock accuracy according to different applications. When measurement data need to match a real event or acquisition systems are not on the same network, absolute time is required. The source of the absolute time can be the Precision Time Synchronization Protocol (PTP) [15] Grandmaster clock, a Network Time Protocol (NTP) [29] server or a Simple Network Time Protocol (SNTP) [30] server.

We synchronize the data by using a software strategy based on the customized clock synchronization scheme referenced from the NTP for the reasons below.

1) The NTP and SNTP can achieve millisecond-level synchronization, which is not sufficient for our requirements. Our moCap cameras operate at 240 $Hz$, and that frequency is not stable. The customized scheme can achieve microsecond-level synchro-

nization based on the C++ chrono module. The chrono module allows us to build a synchronization implementation method that more precisely fits the requirements.

2) The synchronization mechanisms of the PTP and NTP are different. The PTP is designed for correcting a clock on a slave device according to the master. Thus, the clock offset is computed on the slave. The whole procedure requires four synchronous messages, which is stable and secure but has a high time cost. The NTP needs only two synchronous messages, and the final offset is computed on the chosen machine, which is more flexible.

3) The NTP transmits messages using UDP/IP. To increase the stability and decrease the complexity of the synchronization procedure, synchronization is performed once using TCP/IP before the start of each trial.

As Fig. 3.4 shows, the synchronization starts after the start signal of the trial. The server sends the local timestamp T1 to the client. When the client receives T1, it immediately records the local timestamp T2. After waiting for 50 ms, the client records the local timestamp T3 and sends it to the server. After receiving the message, the server records the local timestamp T4. These four timestamps can be used to calculate the time difference of the local time between the server and the client as follows:

$$\begin{cases} \delta &= (T_4 - T_1) - (T_3 - T_2) \\ \theta &= (T_2 - T_1) - \frac{\delta}{2} \\ &= \frac{(T_2 - T_1) + (T_3 - T_4)}{2} \end{cases} \tag{3.1}$$

where $\delta$ is the round-trip delay, $\theta$ is the time offset between the client and the server, and the time offset is the summary of the average time delay and the time error. Thus, the timestamp of the client on the timeline of the server $T_s$ can be backtracked as

$$T_s = T_c + \theta, \tag{3.2}$$

where $T_c$ is the timestamp of the client on the timeline of the client.

The timestamp on the client is tagged on the data of the TOF camera, which operates at 30 frames per second. The backtracked $T_s$ may not match any data of the moCap cameras due to the difference in the absolute time between the two kinds of samples, as shown in Fig. 3.5. Therefore, we use a nearest-search method (Alg:1) to determine which

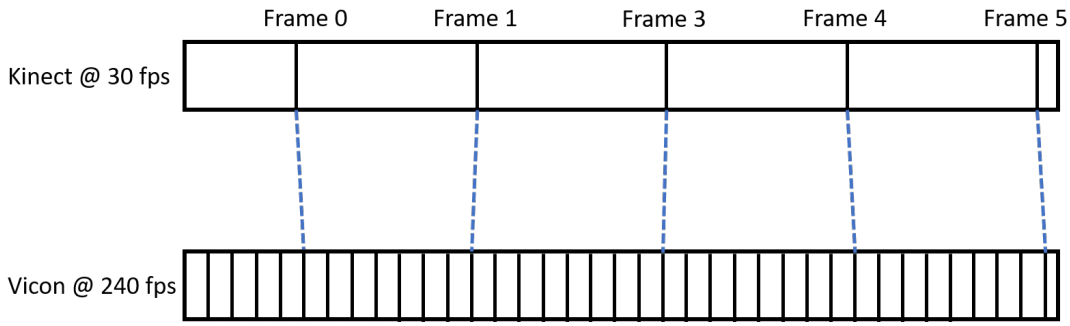



Figure 3.5: Data alignment between the data of the TOF camera and the moCap system

frame of the marker location matches a given frame of the depth map. The method is given a timestamp that is read from a binary file, an offset and marker data that is read from the csv file. A specific frame from the Kinect is matched with the nearest frame from the moCap system. The program traverses all of the binary file, which is classified by subject and motion, obtains the marker locations from the nearest frame, and then writes them as a $h5py$ raw dataset.

After we obtain the $h5py$ raw dataset, we perform data cleaning. In the raw dataset, some markers do not have value because of occlusion or out-of-scope issues during data collection. If such data were kept and used in training the neural network, it would mean that the loss could not be backpropagated. We remove the invalid data by traversing all of the data and checking whether each piece of data has the value $NaN$. Eventually, we have a clean dataset as described in 3.1.

**Algorithm 1:** Nearest search for the timestamp check

**Data:** Timestamp T, Offset O, Data from csv file D

**Result:** position, t

```
previous time PT := 0;
previous item PI := None;
foreach row in D do
    item I := row;
    read time step t from row;
    if T + O < t then
        if T + O - P < t - T - O then
            for pos ← 2 to length(PI) with step 3 do
                position[marker index, 0] := PI[pos];
                position[marker index, 1] := PI[pos+1];
                position[marker index, 2] := PI[pos+2];
            end
        else
            for pos ← 2 to length(I) with step 3 do
                position[marker index, 0] := I[pos];
                position[marker index, 1] := I[pos+1];
                position[marker index, 2] := I[pos+2];
            end
        end
        break;
    end
    PT := t;
    PI := I;
end
return position, t;
```

# Chapter 4

# Pose Estimation Network

In this chapter, we introduce the hybrid models that estimate the 3D human pose from a depth map. We first introduce our designed preliminary model in Section 4.1. Then, we introduce an improved model for increasing the estimation accuracy and avoiding the occlusion problem faced by the preliminary model in Section 4.2. Then, in Section 4.3, we describe the joint loss that can be used to make the backpropagation update the weights with multiple losses at the same time.

## 4.1 Preliminary Model

The preliminary model is designed based on the pose estimation model in the state-of-the-art paper [14] with a slight modification. To estimate the 3D human pose from a depth map, a preliminary train of thought is to obtain the $x$ and $y$ coordinates as a 2D pose estimation task. Then, the corresponding depth values of the pixels of the joints can be retrieved and used as the z-coordinates. Next, we introduce the preliminary model.

### 4.1.1 Model Architecture

[14] proposed a network architecture containing a backbone network to extract features and an adaptive weighting regression (AWR) method to aggregate dense representations. This model uses depth offset maps as the in-plain feature, which is essentially computationally expensive. To make the architecture work with the 2D internal probability heat map representation instead of the depth offset maps, we modify the AWR to soft-argmax and include the dot product for the generation of z-coordinates. The model's details are described below.

We designed the model from the inspiration of the 2D pose estimation and the characteristics of the depth map, as shown in Fig. 4.1. The model has a main pathway to estimate the 2D heat map of each joint. We take the dot product of the depth image and each upsampled heat map to obtain the approximate z-coordinates. The z-coordinates generated by this operation are greatly impacted by points that occlude the target joint; thus, the model is suitable for application scenarios without occlusion. The x-coordinates and y-coordinates are computed by soft-argmax. After the three coordinates are concatenated, a linear layer maps them into real-world coordinates.

### 4.1.2 Feature Extractor

The purpose of our feature extractor is to extract high-dimensional features of size ($N \times 256 \times 64 \times 64$) from a depth image of size ($N \times 1 \times 256 \times 256$), where $N$ is the batch size. The architecture of the feature extractor is shown in Fig. 4.2. It is composed of

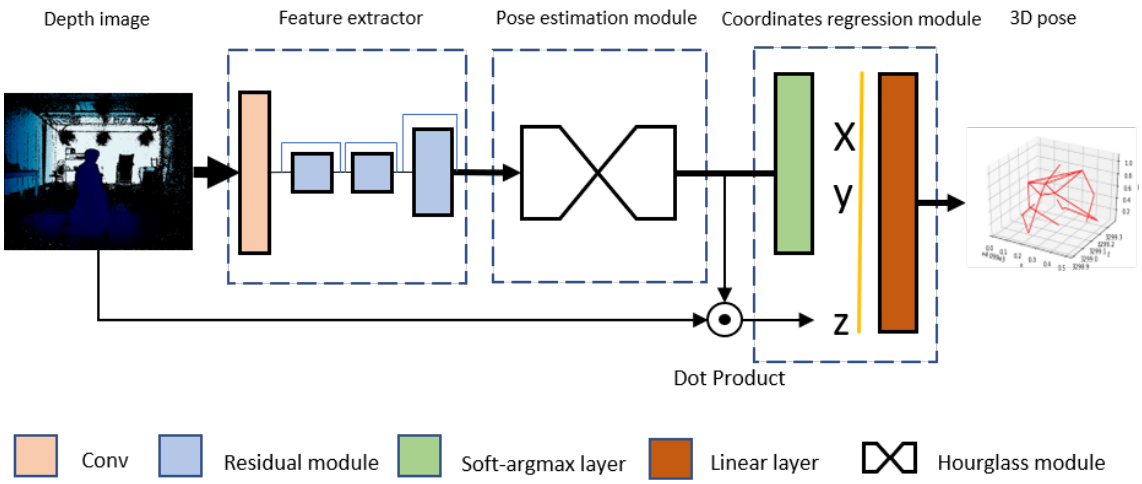



Figure 4.1: Overall network architecture diagram of the preliminary model.

a convolutional layer followed by three residual modules. There is only one maxpooling operation between the first residual module and the second module to shrink the feature size.

### 4.1.3 Pose Estimation Module

Our pose estimation module is composed of an hourglass module that was designed relative to the encoder-decoder architecture and a convolutional layer for restructuring the features. The output of the pose estimation module is a set of heat maps of joints. Fig. 4.3 shows the details of the pose estimation module.

We build a 4-stage hourglass module with 14 residual blocks, 4 maxpooling layers and 5 upsampling layers. The output feature's size is $(N \times 256 \times 64 \times 64)$, which is the same as that of the output of our feature extractor. To generate the heat maps of the joints, we design convolutional layers behind the hourglass module to mutate the channel of the feature from 256 to the number of joints $K$ so that the final feature has a size of $(N \times K \times 64 \times 64)$. Each channel of the feature is a heat map of a corresponding joint. In the preliminary model, the heat maps correspond only to the x and y dimensions of the depth image.

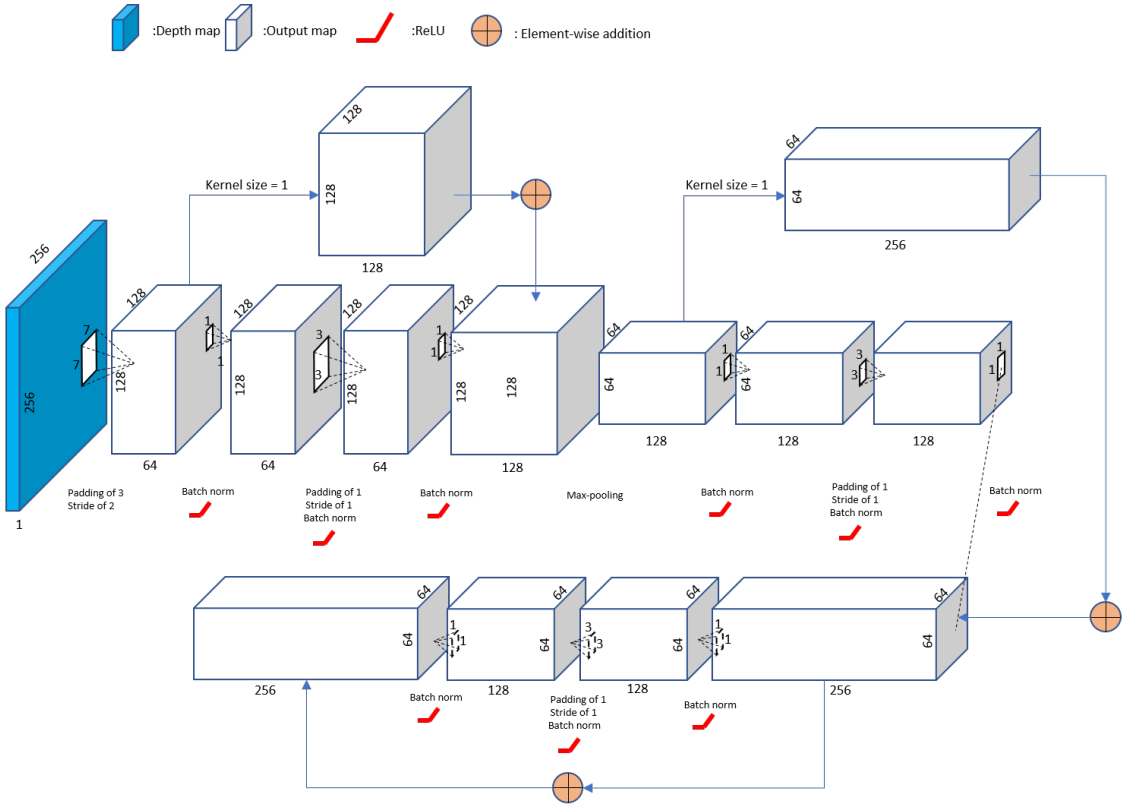
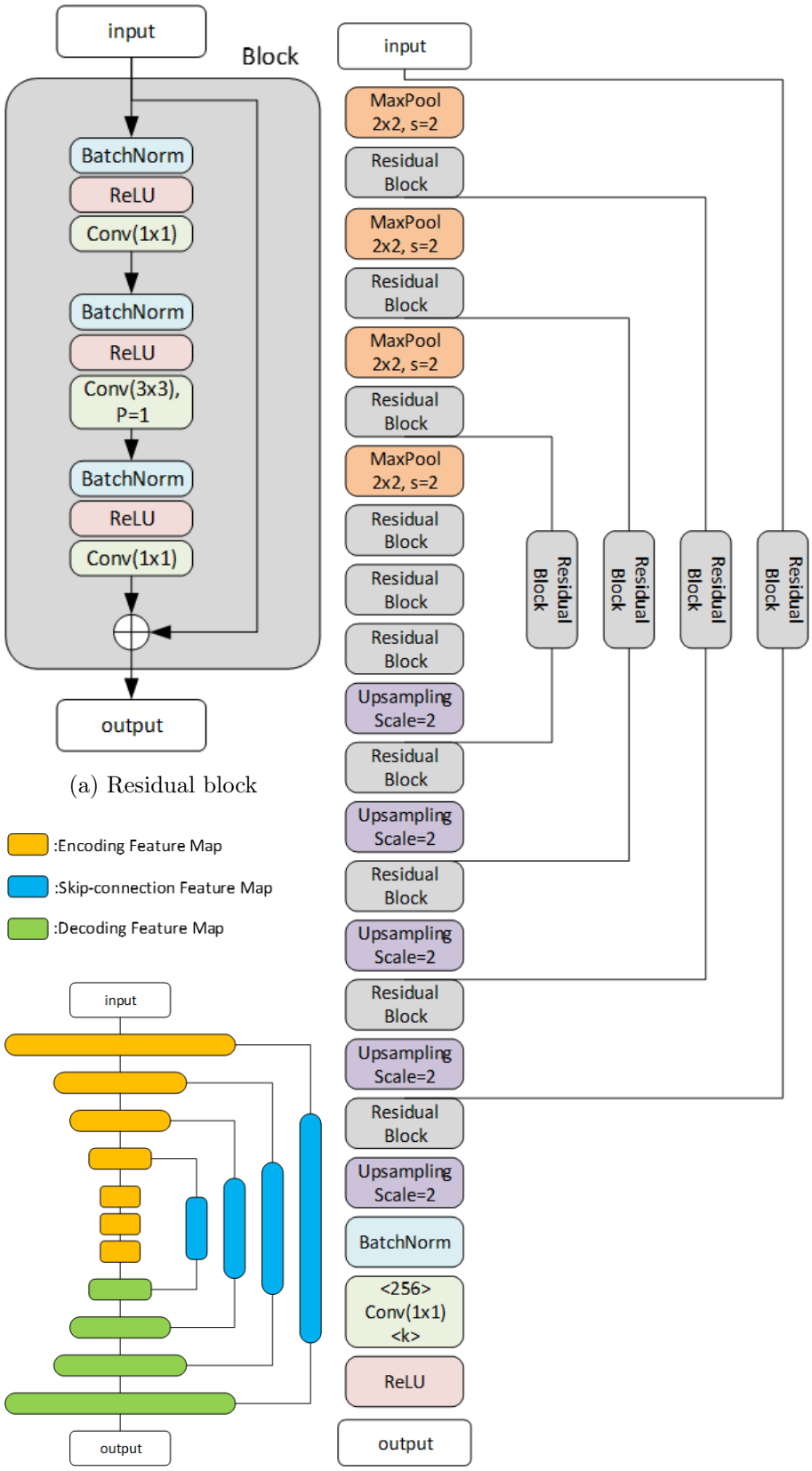



Figure 4.2: Architecture of our feature extractor. Each box is a feature map.




(a) Residual block

(b) Flat view of the feature map

(c) Structure of the 4-stage hourglass module



Figure 4.3: Architectural detail of a single-column heat map generator

### 4.1.4 Coordinate Regression Module

To obtain the estimated z-coordinates, we upsample the heat maps from the pose estimation module by the nearest interpolation from a size of $64 \times 64$ to the size of the depth map $256 \times 256$. Then, we manipulate the dot products on each heat map and the depth map to obtain the estimated z-coordinates. The soft-argmax layer computes the x- and y-coordinates from the heat maps. Thus far, the x- and y-coordinates are values that are normalized to $(-1, 1)$, and the z-coordinates are the expectation of the depth. To map them into the space of the real world, we concatenate the x-, y- and z-coordinates and linearly transform them with a linear layer.

## 4.2 An Improved Model

The preliminary model provides a space-efficient solution for 3D human pose estimation. Nevertheless, the estimation accuracy is not satisfactory since the model has a small capacity and the depth information is concealed by the foreground points. We propose an improved model in this section to obtain a satisfactory estimation result and solve the occlusion problem.

### 4.2.1 Model Architecture

The main part of our module is based on the hourglass module introduced in the related work. The network first extracts the input depth map as high-dimensional features through a convolution layer and three residual modules, which is the same as the preliminary model. Then, these high-dimensional features pass through three parallel hourglass structures with the same structure to generate three heat maps. These three heat maps are used in combination to calculate the expected value along the corresponding coordinate axis following the equation 4.1. These three heat maps are transformed into three expectations along different coordinate axes through a soft-argmax layer. Finally, these three expectations are linearly mapped to the world coordinate system. The overall network architecture diagram is shown in Fig. 4.4.

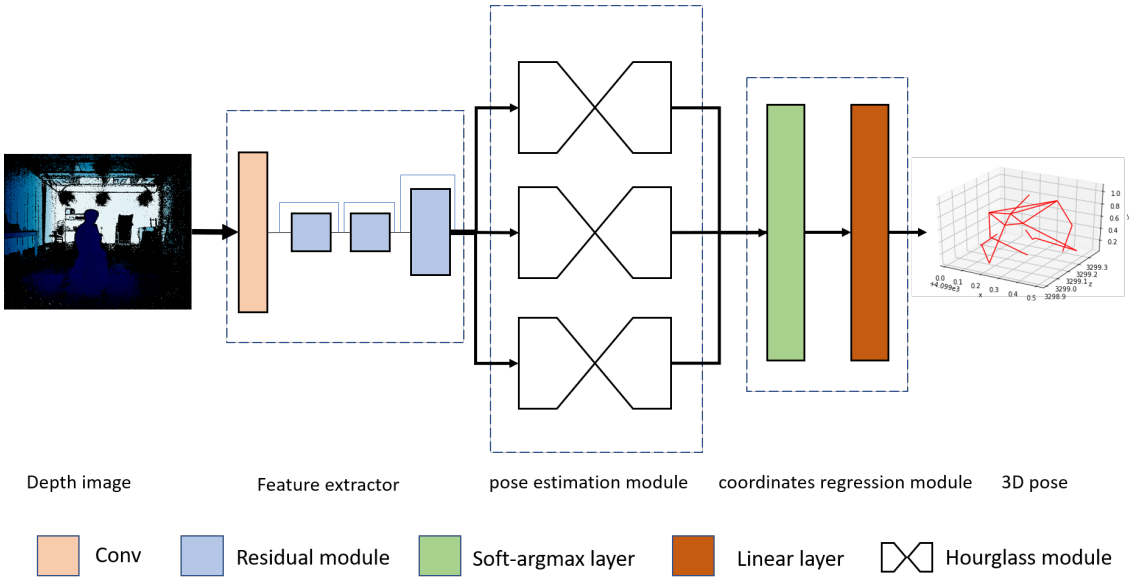



Figure 4.4: Overall network architecture diagram.

### 4.2.2 Pseudo-3D Soft-argmax

As introduced in the related work, three heat maps from three views can represent a voxel heat map. In [33], the argument of the maxima $\mu_x$ and $\mu_y$ are computed using only $\hat{H}^{(xy)}$ because the input image does not include depth information. The accuracy of the heat maps $\hat{H}^{(xz)}$ and $\hat{H}^{(zy)}$ is relative to only the z-coordinates.

However, in addition to $\mu_x$ and $\mu_y$ calculated from $\hat{H}^{(xy)}$, the expectations of $\hat{H}^{(xz)}$ along the x-axis and $\hat{H}^{(zy)}$ along the y-axis are $\mu_x$ and $\mu_y$, respectively. If we compute the argument of the maxima as the average of the $\mu$ from two heat maps in the supervised network, this value can ultimately improve the accuracy and relevance of the heat maps. In theory, the expectations on the same axis of the two heat maps should be the same. We strengthen the connection between the three heat maps by computing the arithmetic mean of the expectation as:

$$\begin{aligned} \mu_x &= \tfrac{1}{2}(< \hat{H}^{(xy)} \cdot T_x > + < \hat{H}^{(xz)} \cdot T_x >) \\ \mu_y &= \tfrac{1}{2}(< \hat{H}^{(zy)} \cdot T_y > + < \hat{H}^{(xy)} \cdot T_y >) \\ \mu_z &= \tfrac{1}{2}(< \hat{H}^{(zy)} \cdot T_z > + < \hat{H}^{(xz)} \cdot T_z >) \end{aligned} \tag{4.1}$$

**Algorithm 2:** Gaussian distribution heat map generator

**Input:**mean $\mu$, $size$, $\sigma$

**Output:**Gaussian-distributed heat map $G$

Denote the first value $F$ as $-(size-1)/size$

Denote the last value $L$ as $(size-1)/size$

Generate the coordinate matrix $M$ of $size$ by a 2D linspace from $F$ to $L$

Compute the Gaussian distribution matrix $G$ with $\exp(-(M-\mu)^2/(2\sigma^2))$

Normalize the matrix $G$ by dividing by the summation of the matrix itself.

## 4.3 Joint Loss

For the regression tasks in deep learning, the Euclidean distance is widely used to compute the distance between the target values and the predictions. Thus, a straight solution is used to measure the distance between the estimated coordinates and the ground truth. Minimizing the distance in the training process eventually brings the estimated results close to the ground truth. The Euclidean loss function is given in Equation 4.2 below:

$$\mathcal{L}^{euc} = \sqrt{\sum_{i=1}^{K}(\hat{y}_i - y_i)^2}, \tag{4.2}$$

where $\hat{y}$ represents the estimated value and $y$ represents the target value.

The Jensen-Shannon divergence (JS) [7] measures the similarity between two probability distributions. To ensure that the heat maps in the network can be constrained to the Gaussian probability distribution, we calculate the pixelwise Jensen-Shannon divergences between the network-generated heat maps and the ideal heat maps computed from the coordinates. We compute the Gaussian distribution heat maps from the ground-truth coordinates following Algorithm 2 below.

Before generating the ground-truth heat maps, the paramount prepossessing step on the ground-truth coordinates is normalization. Each dimension of the coordinates is normalized to the open interval $(-1, 1)$. For the preliminary model, only the x and y dimensions are considered to generate heat maps; thus, the JS loss function can be

described as follows:

$$\begin{aligned} \mathcal{D}_{KL}(p \parallel q) &= \sum_{i=1}^{N} p(x_i) \cdot (\log p(x_i) - \log q(x_i)) \\ \mathcal{L}^{JS} &= 0.5 \cdot \mathcal{D}_{KL}(p \parallel \tfrac{p+q}{2}) + 0.5 \cdot \mathcal{D}_{KL}(q \parallel \tfrac{p+q}{2}) \end{aligned} \tag{4.3}$$

where $p$ is the heat maps generated from the network and $q$ is the ground-truth heat maps that are generated by performing Algorithm 2.

For the preliminary model, the total loss used is determined by Equation 4.4:

$$\mathcal{L} = \mathcal{L}^{euc} + \mathcal{L}^{JS}_{xy}. \tag{4.4}$$

For the improved model we introduced above, we compute the JS loss for each view of the heat maps. Thus, the total loss used is determined by Equation 4.5:

$$\mathcal{L} = \mathcal{L}^{euc} + (\mathcal{L}^{JS}_{xy} + \mathcal{L}^{JS}_{zy} + \mathcal{L}^{JS}_{xz}). \tag{4.5}$$

# Chapter 5

# Experiments and Results

In this chapter, we present the details of the experiments of our work to verify the proposed models on the side view of the ITOP dataset and Kivi dataset. In Section 5.1, we report the preparation of the data process for compatible loading into the network. We introduce the environment configuration in Section 5.2 and the evaluation criterion in Section 5.3. We report the details of the training parameters in Section 5.4, including the parameters of the optimizer, the initialization of the network, the pivotal parameters for the internal feature maps and the generated poses. Then, in Section 5.5, we describe the evaluation of 3D pose estimation for two models on the side view of the ITOP dataset

and Kivi dataset and the accelerated results using ONNX in the Caffe 2 and TensorRT frameworks.

## 5.1 Data Preparation

We evaluate our models on two datasets: one is the side view of the ITOP dataset, and the other is our proposed Kivi dataset. We introduce the preparatory procedures for the two datasets as follows:

- ITOP dataset: For the depth images, the original size is $240 \times 320$, which does not fit our model's input size requirement, which should be $256 \times 256$. We crop 32 columns horizontally on each side of the images and add 8 rows of zeros above and below the images. For the labels, the manipulation used to resize the images does

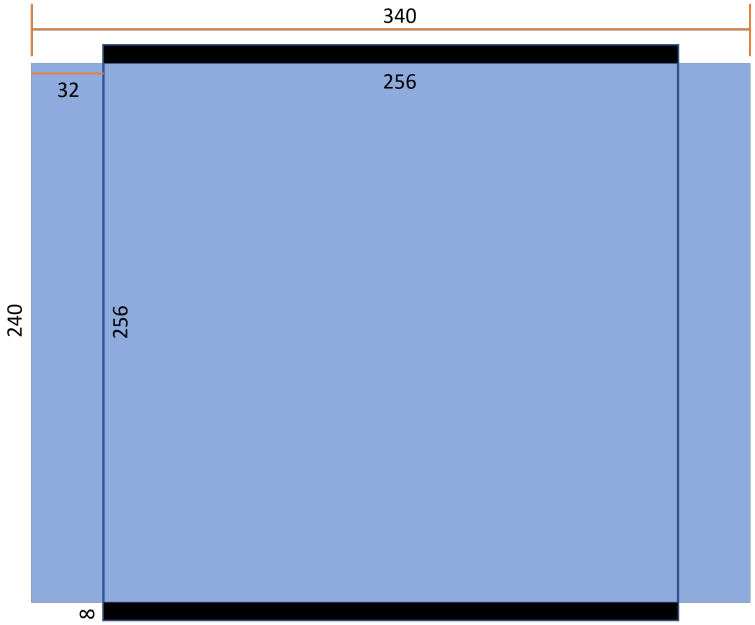



Figure 5.1: The resizing manipulation of the depth map of the ITOP dataset.

not affect the real-world 3D coordinates. In the coordinates of the ITOP dataset, the x- and y-coordinates have a linear relationship with the row and column indexes of the depth map, and the z-coordinate is the depth dimension of the depth map. Based on the above two considerations, we do not mutate the coordinates.

To generate the Gaussian distributed heat maps when computing $\mathcal{L}^{JS}$, we normalize the coordinates to $(-1, 1)$ using the following Equation 5.1:

$$\begin{aligned} X_{normed} &= (X + 0.89)/1.9 \times 2 - 1 \\ Y_{normed} &= (Y + 1.7)/2.7 \times 2 - 1 \\ Z_{normed} &= (Z - 1.8) - 1 \end{aligned} \tag{5.1}$$

- Kivi dataset: For the depth images, the original size is $424 \times 512$. We resize the image to $256 \times 256$ using bicubic interpolation.

  For the labels, similar to the ITOP dataset, the resizing manipulation does not affect the real-world 3D coordinates in the Kivi dataset. However, we use the Cartesian coordinate system in the Kivi dataset, which indicates the positive direction of the Z-axis as being up. To ensure that the coordinate systems of the datasets are consistent, we first convert the coordinates using Equation 5.2 below.

$$\begin{aligned} X_{convert} &= 4.1 - Y \\ Y_{convert} &= Z \\ Z_{convert} &= 3.3 - X \end{aligned} \tag{5.2}$$

  The converted coordinates are eventually used as the ground-truth coordinates.

  We normalize the converted coordinates to $(-1, 1)$ and compute $\mathcal{L}^{JS}$ using the following Equation 5.3:

$$\begin{aligned} X_{normed} &= (X - 1.96)/3.4 \times 2 - 1 \\ Y_{normed} &= (Y + 0.003)/2.4 \times 2 - 1 \\ Z_{normed} &= (Z - 0.78)/4.2 \times 2 - 1 \end{aligned} \tag{5.3}$$

## 5.2 Environmental Configuration

We perform data cleaning and network training on the same platform, which has the following detailed configuration: The operating system on the platform is 64-bit Ubuntu 18.04.3 LTS. The machine has an Intel i7-9700K CPU, 62.7 gigabytes of memory and dual GeForce GTX 1080 Ti GPUs. To implement and evaluate the neural network, we

deploy the programs with PyTorch 1.3.1 on Python 3.6.9. The CUDA compilation tool version is 10.1.243.

## 5.3 Evaluation Criterion

In this section, we introduce the evaluation method we use to gauge the precision. Suppose we have $N$ subjects $S_1, S_2, ..., S_n$ and that each subject has $K$ joints $J_1, J_2, ..., J_k$ to indicate the pose. The accuracy of each joint is restricted by a threshold, as shown in Fig 5.2. We compute the Euclidean distance between the coordinates of the estimated joint $(\hat{x}, \hat{y}, \hat{z})$ and those of the ground truth $(x, y, z)$. If the distance is smaller than the prescribed threshold, we record the estimated joint as correct. The overall mean average precision (mAP) is defined in Equation 5.4:

$$mAP_k = \frac{\sum_{n=1}^{N} \epsilon(threshold - \sqrt{(\hat{x_k} - x_k)^2 + (\hat{y_k} - y_k)^2 + (\hat{z_k} - z_k)^2})}{N}, \tag{5.4}$$

where $mAP_k$ is the mAP of the $k$th joint and $\epsilon()$ is the step function. Then, the total mAP of all the joints is given by Equation 5.5:

$$mAP = \frac{\sum_{k=1}^{K} mAP_k}{K}. \tag{5.5}$$

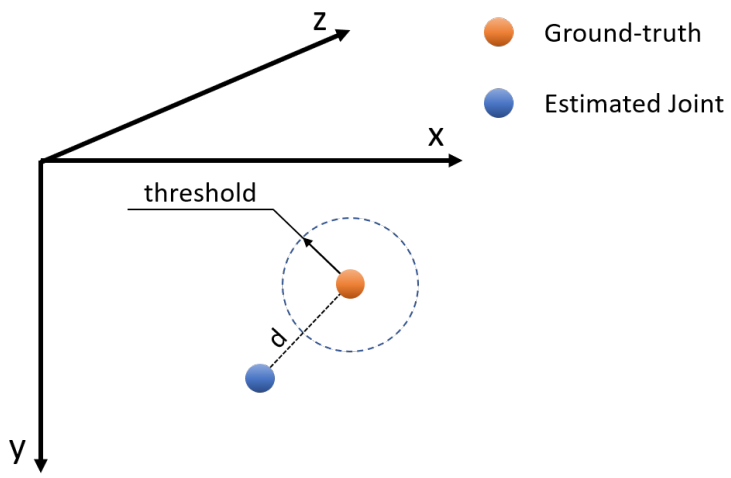



Figure 5.2: The threshold determines whether the estimated result is acceptable. If $d < threshold$, we say the estimated joint is acceptable.

## 5.4 Training Parameters

In this section, we describe the parameters we establish for training the proposed model. We use the Adam optimizer [21] with learning rates $\alpha = 0.0005$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ and a batch size of 16 for a mini-batch. The parameters set for the optimizer do not affect the estimation precision but impact the convergence speed when training, as recommended by [21]. We initialize the weights in the convolutional layer and the linear layer with the Kaiming initialization [11]. For the ground truth of the heat maps, we set $\sigma$ as 1. The number of classes indicates how many joints are used to represent the skeleton in the specific dataset. This parameter affects the shape of the heat map tensor and the number of neurons in the final linear layer. All the training parameters are shown in Table 5.1.

| Parameters | Preliminary model on ITOP | Improved model on ITOP | Preliminary model on Kivi | Improved model on Kivi |
|---|---|---|---|---|
| Optimizer | Adam | Adam | Adam | Adam |
| Learning rate $\alpha$ | 0.0005 | 0.0005 | 0.0005 | 0.0005 |
| $\beta_1$ | 0.9 | 0.9 | 0.9 | 0.9 |
| $\beta_2$ | 0.999 | 0.999 | 0.999 | 0.999 |
| $\sigma$ | 1 | 1 | 1 | 1 |
| Num of classes | 15 | 15 | 18 | 18 |

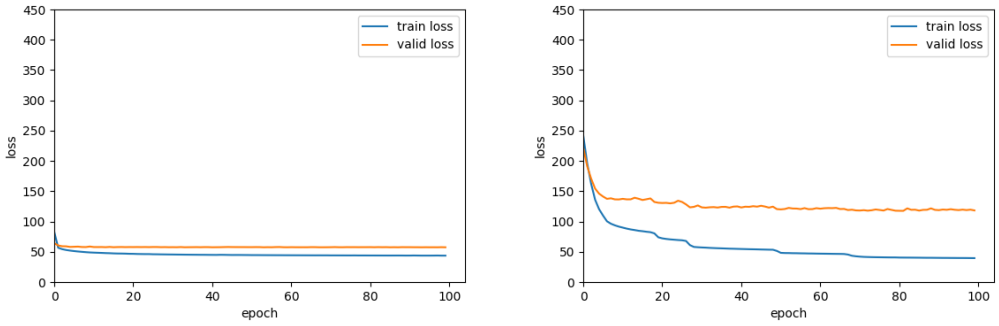
Table 5.1: The training parameters for the four experiments.

## 5.5 Experiments

In this section, we present experiments that include the training and evaluation of two models separately on the two discussed datasets.

### 5.5.1 Models on the ITOP dataset

To compare the performance of the proposed approach with state-of-art methods, we train our preliminary model and our improved model on the side view of the ITOP dataset and gauge the $mAP$ with a threshold of 10 cm.




(a) The loss curve on the preliminary model (b) The loss curve on the improved model

Figure 5.3: The loss of training and validation on the ITOP side-view dataset.

The loss curve in Fig 5.3 shows that the preliminary model converges on the validation set in 20 epochs. However, continuous training results in overfitting. The improved model can continue learning until approximately 70 epochs. We compute the $mAP$ on each part of the body and on the full body and then plot the $mAP$ for different error thresholds in Fig 5.5. When we increase the error threshold from 0 meters to 0.8 meters, the $mAP$ of each joint increases and then becomes flat. By comparing the curves from the two models, we find that the improved model has curves that increase faster. We compare the $mAP$ at 10 cm with the state-of-the-art value, which is shown in Table 5.2. The results show that our improved model beats the other approaches on the $mAP@10cm$ value for the full body, which is due to the significant improvement of the lower body precision.

To display the estimated results, we design a user interface, as shown in Fig 5.4.

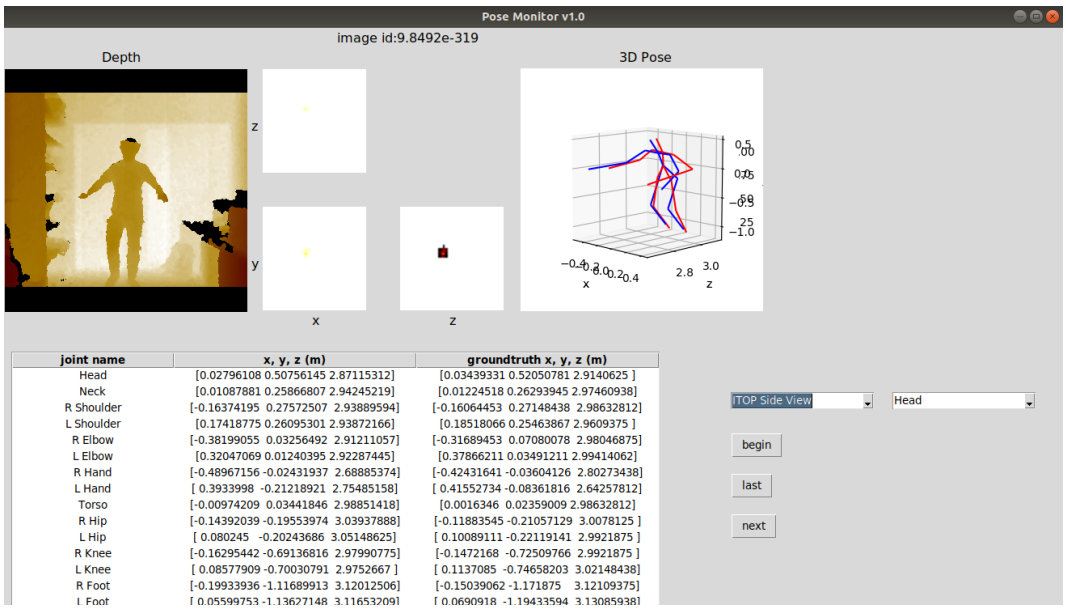
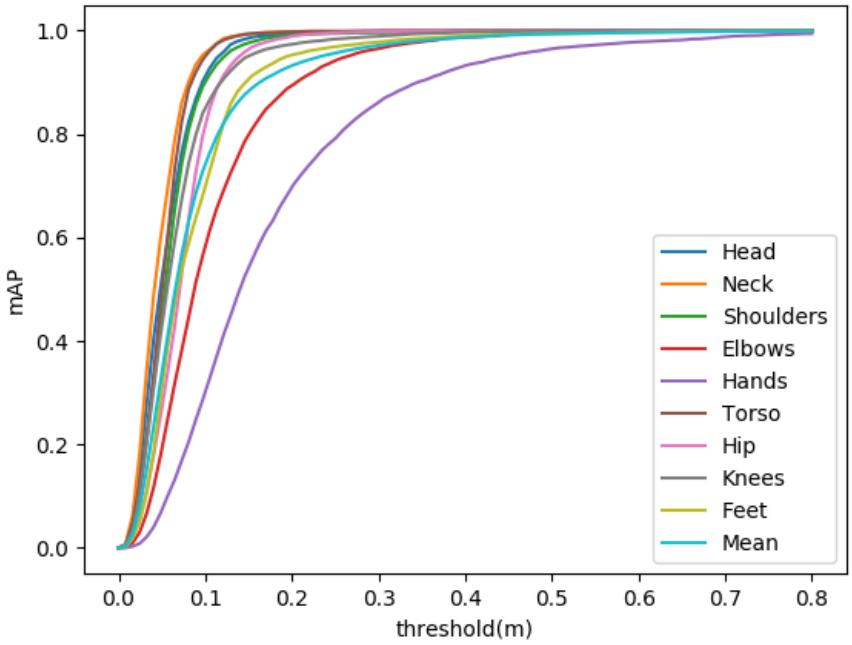
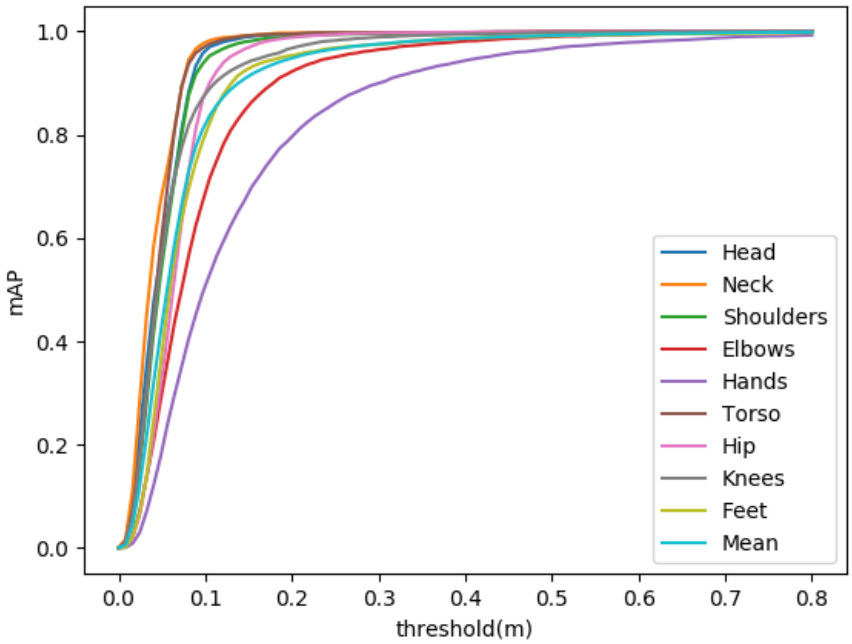



Figure 5.4: Pose monitor designed to display the estimated result.




(a) The $mAP$ for different error thresholds on the preliminary model




(b) The $mAP$ for different error thresholds on the improved model

Figure 5.5: The percentage of successful joints for different error thresholds on the side view of the ITOP dataset.

| Body Part | RTW | RF | IEF | VI | Preliminary | Improved |
|---|---|---|---|---|---|---|
| Head | 97.6 | 63.8 | 96.2 | **98.1** | 83.93 | 96.44 |
| Neck | 95.8 | 86.4 | 85.2 | 97.5 | 89.01 | **97.81** |
| Shoulders | 94.1 | 83.3 | 77.2 | **96.5** | 76.36 | 94.43 |
| Elbows | **77.9** | 73.2 | 45.4 | 73.3 | 46.33 | 68.98 |
| Hands | **70.5** | 51.3 | 30.9 | 68.7 | 22.36 | 50.75 |
| Torso | 93.8 | 65.0 | 84.7 | 85.6 | 88.33 | **97.00** |
| Hips | 80.3 | 50.8 | 84.7 | 85.6 | 68.70 | **88.24** |
| Knees | 68.8 | 65.7 | 81.8 | 69.0 | 72.91 | **87.97** |
| Feet | 68.4 | 61.3 | **80.9** | 60.8 | 60.25 | 80.20 |
| Upper Body | **84.8** | 70.7 | 61.0 | 84.0 | 57.88 | 77.82 |
| Lower Body | 72.5 | 59.3 | 82.1 | 67.3 | 70.29 | **87.12** |
| Full Body | 80.5 | 65.8 | 71.0 | 77.4 | 63.67 | **82.16** |

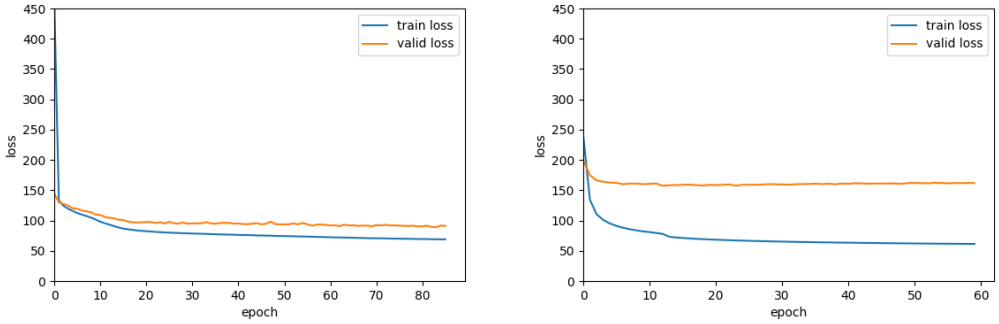
Table 5.2: Comparison between our models and the state-of-the-art methods for the $mAP$ at the 10 cm error threshold. The precision data of the state-of-the-art methods are from [9]

### 5.5.2 Models on the Kivi dataset

We also trained our models on our proposed dataset, which has 3 more joints and includes more noise than the ITOP dataset; thus, it should be more challenging.

The loss curves (Fig 5.6) show that both models can converge in 20 epochs. The mAP at 10 cm over different thresholds is shown in Fig 5.7. Similar to the ITOP dataset, the models have the lowest mAP on the hand (wrist) joint. This is because the hand (wrist) joint has a low dependency on the other joints and occupies few pixels in the picture.

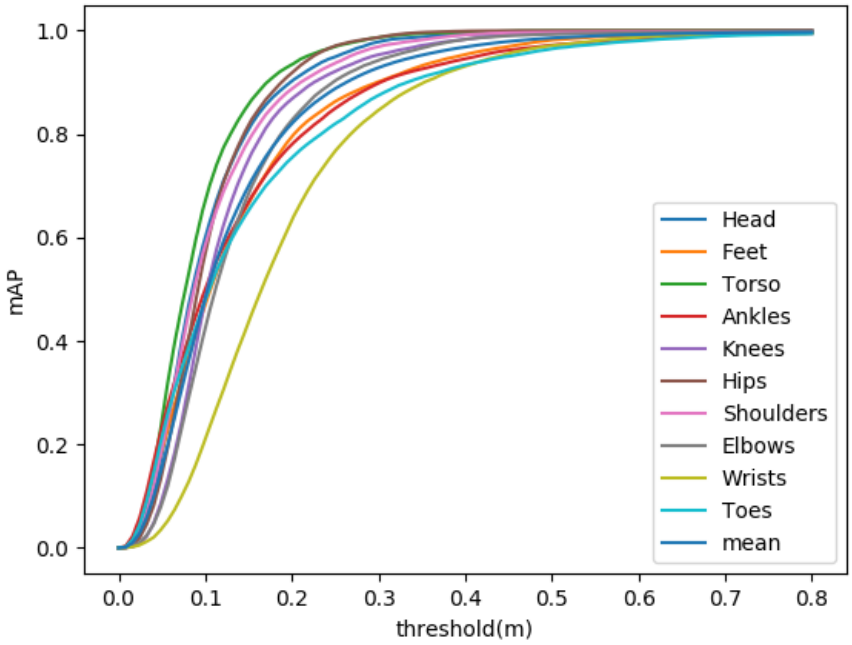
We digitize the mAP at 10/15/20 cm in Table 5.3 to observe the change in accuracy of the two models for different error thresholds. The accuracy increases rapidly between 10 cm and 20 cm. The mAP at 20 cm is approximately 53% higher than the mAP at 10 cm.



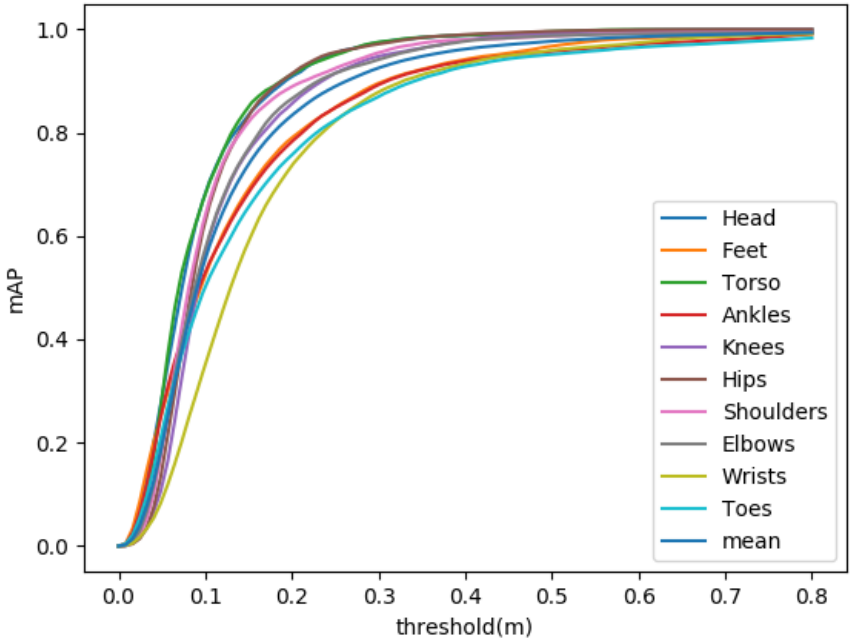

(a) The loss curve of the preliminary model (b) The loss curve of the improved model

Figure 5.6: The loss of training and validation on the Kivi dataset.

Fig. 5.8 shows the successful cases when estimating the 3D human pose on the Kivi dataset, while Fig. 5.9 shows the failure cases. In failure case (a), the model incorrectly estimates the y-axis coordinates of the upper body. We can see that the estimated upper body is slightly higher than the ground truth. This is due to the small number of ground motions in the dataset; the model implicitly thinks that the upper body of the person should appear higher. For the second failure case (b), we can see that the right hand is not estimated well. This is because of the occlusion in the depth image. In the third failure case (c), the estimation fails on two hand positions and the z-coordinates of other




(a) The $mAP$ for different error thresholds on the preliminary model



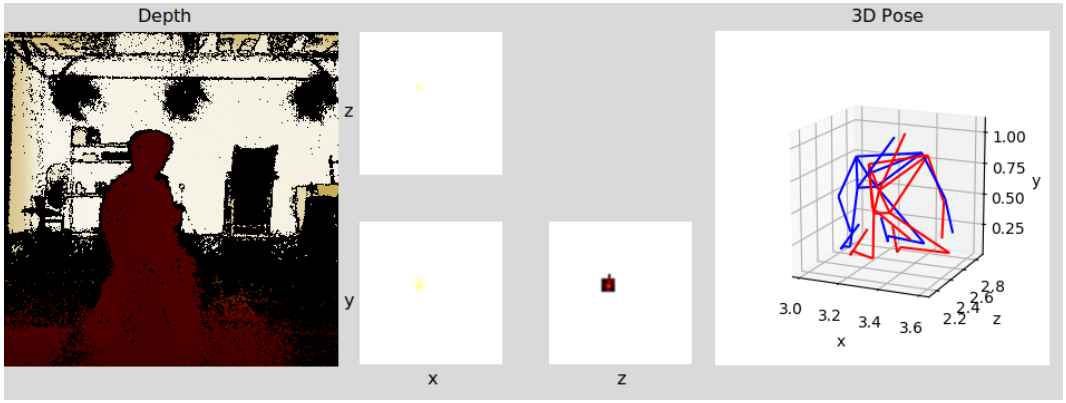

(b) The $mAP$ for different error thresholds on the improved model

Figure 5.7: The percentage of successful joints for different error thresholds on the Kivi dataset.

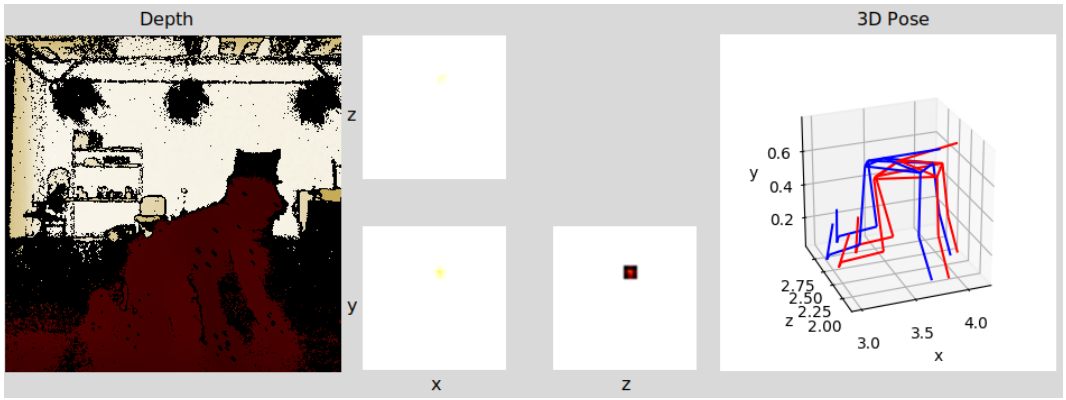
| Model | Preliminary | | | Improved | | |
|---|---|---|---|---|---|---|
| Threshold | 10 cm | 15 cm | 20 cm | 10 cm | 15 cm | 20 cm |
| Head | 58.21 | 79.54 | 89.43 | 66.76 | 82.47 | 90.20 |
| Feet | 45.50 | 65.26 | 78.24 | 51.04 | 67.96 | 78.18 |
| Torso | 65.31 | 84.64 | 92.86 | 66.66 | 83.98 | 90.68 |
| Ankles | 48.68 | 65.54 | 76.98 | 51.80 | 67.26 | 77.33 |
| Knees | 45.50 | 73.43 | 85.93 | 54.51 | 75.48 | 84.69 |
| Hips | 54.67 | 80.32 | 90.95 | 60.90 | 81.93 | 90.72 |
| Shoulders | 56.82 | 77.40 | 87.95 | 62.19 | 81.10 | 88.34 |
| Elbows | 40.42 | 66.71 | 81.46 | 55.88 | 75.77 | 85.87 |
| Wrists | 19.62 | 41.61 | 61.10 | 33.55 | 57.22 | 72.24 |
| Toes | 46.48 | 64.03 | 74.45 | 48.95 | 64.50 | 74.75 |
| Mean | 46.61 | 68.49 | 80.91 | 53.95 | 72.72 | 82.51 |

Table 5.3: The mAP of two models on the Kivi dataset for three thresholds.

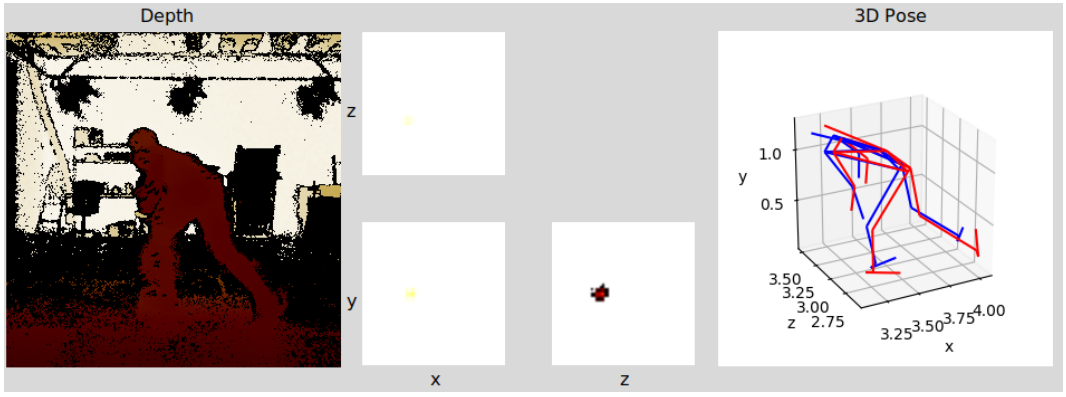
joints.




(a)

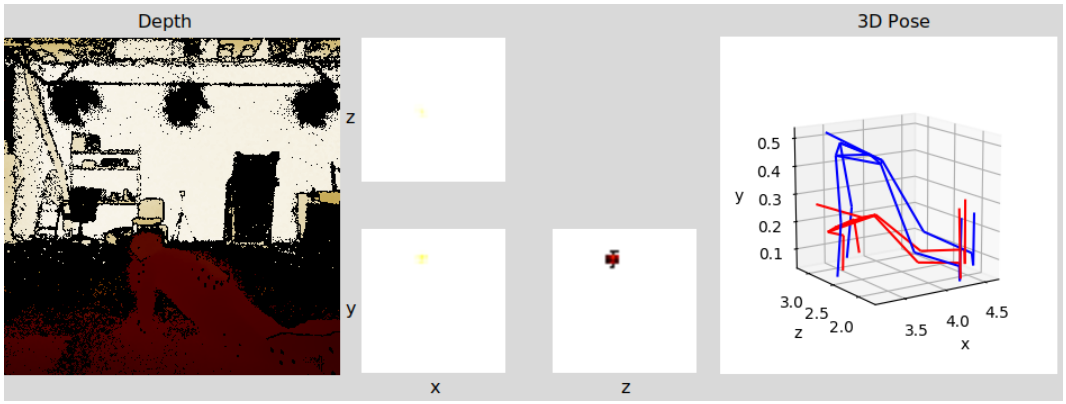



(b)

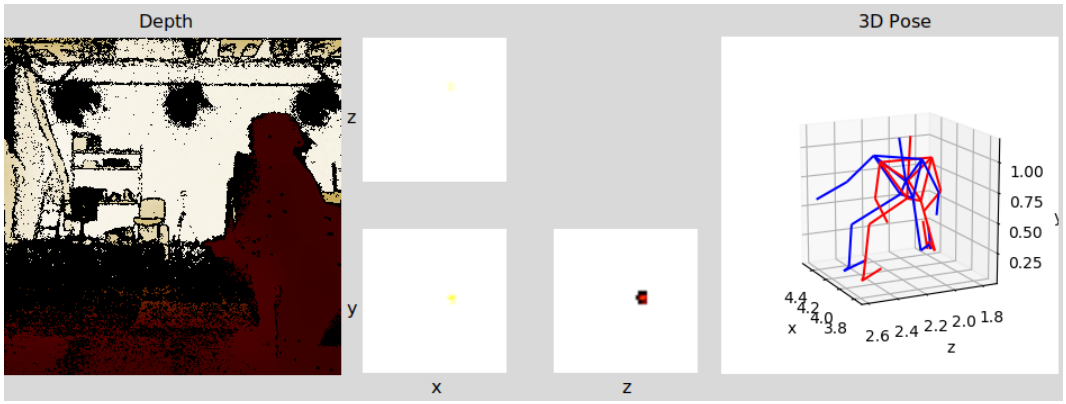



(c)

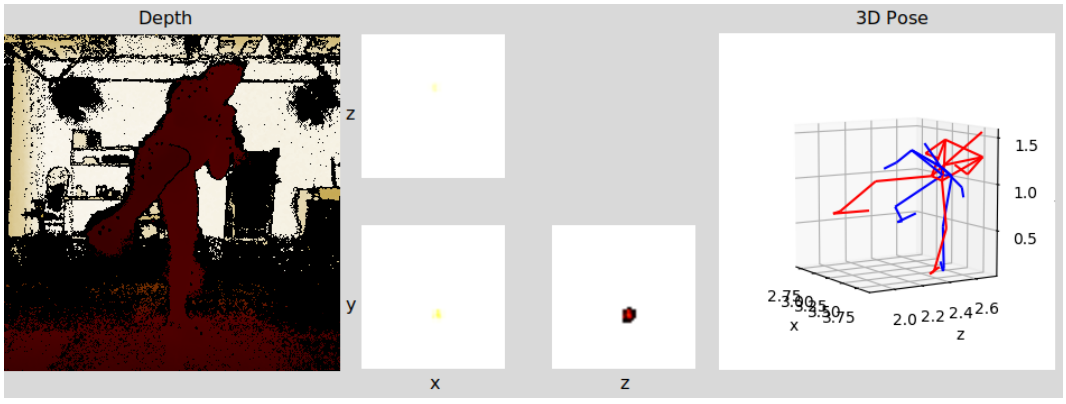
Figure 5.8: A successful estimation result on the Kivi dataset. From left to right are the depth image, the heat map of the head joint, and the 3D pose. The red pose represents the ground truth, and the blue pose represents the estimated result.




(a)




(b)




(c)

Figure 5.9: Failure cases on the Kivi dataset. From left to right are the depth image, the heat maps of the head joint, and the 3D pose. The red pose represents the ground truth, and the blue pose represents the estimated result.

### 5.5.3 Runtime Analysis

For 3D human pose estimation in application scenarios, the model should run in a real-time environment. We test our improved model with the three different frameworks to gauge the runtime frames per second and see whether the model can run in real time. We load the model into the GPU's memory in advance and load the weights we had trained. We set the batch size to 1 when testing so that the model runs on only one GPU and processes only one frame of input data at a time. We use this method to simulate the performance of the model in real-time data processing online. The data used in this experiment are all from the test set of our Kivi dataset, which has a total of 14,277 frames. The frame rate is calculated after all the data are used and is divided by the total running time. All three experiments are run on an Nvidia 1080Ti GPU.

The first framework is PyTorch [36], where we design, train and validate our model. Many factory-level applications do not deploy AI applications on PyTorch because it does not make any optimizations for deployment. The frame rate of our model on PyTorch is used as the benchmark for this experiment.

The second framework is Caffe2 [17]. Caffe2 integrates many emerging algorithms and models based on Caffe, which strengthens the framework's ability to deploy lightweight hardware platforms, thereby helping developers to deploy AI models on mobile devices and to quickly and accurately handle related analysis tasks. Pytorch's model requires transformation to use Caffe2 for forward inference. There are roughly two ways to perform this transformation. The first way is to rewrite the code of the model in the Caffe2 framework and then load the trained weight parameters into the written model. The second way is to convert the PyTorch model to an intermediate model called Open Neural Network Exchange (ONNX) and assign the weight parameters and then use Caffe2 to read the ONNX model for forward inference. Clearly, the second method is more convenient and faster to deploy.

The third framework is TensorRT [34] SDK, which was developed by Nvidia for optimizing a trained neural network on a GPU. The specific optimization method is shown in Fig. 5.10. For importing the Pytorch models into TensorRT, there are the

same two possible schemes as for importing them into Caffe2: one is to build a neural network directly in TensorRT, and the other is to import the model through the ONNX model.

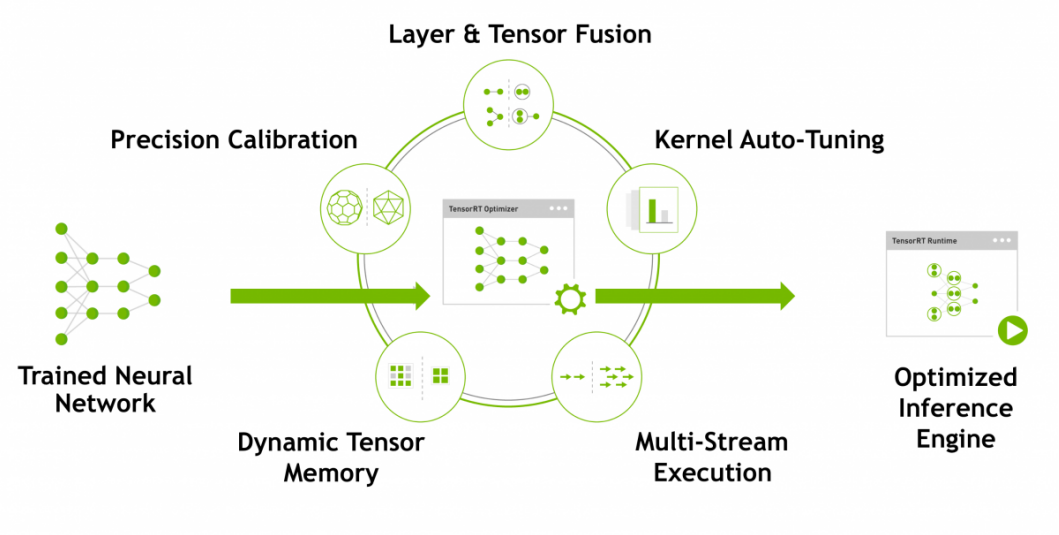



Figure 5.10: Optimization method of TensorRT for neural networks. [34]

Pytorch supports a dynamic computational graph. For example, we do not need to fix the shape and dimension of a certain tensor; it is automatically calculated by Pytorch during training. This feature greatly simplifies the workload of model modification in the design stage. However, when we migrate the model to TensorRT, we must modify all the dynamic computational graphs to static computational graphs. We must check that all the layers we use are supported by TensorRT and ONNX.

Fig. 5.11 shows the frame rate for the three frameworks we introduced above. A well-functioning Kinect can achieve a transmission frame rate of 30 fps or 60 fps. From the experiments, we find that due to the bloated model structure and the large number of parameters, real-time operation cannot be achieved on PyTorch or Caffe2 (only a single Nvidia 1080Ti GPU is used as the running configuration). On TensorRT, even if Kinect transmits a depth map at a maximum frame rate of 60 fps, the model can generate an estimated pose in real time.

This experiment is independent of the dataset because the inference efficiency of the model is related only to the model. Therefore, we can compare the 3D human pose

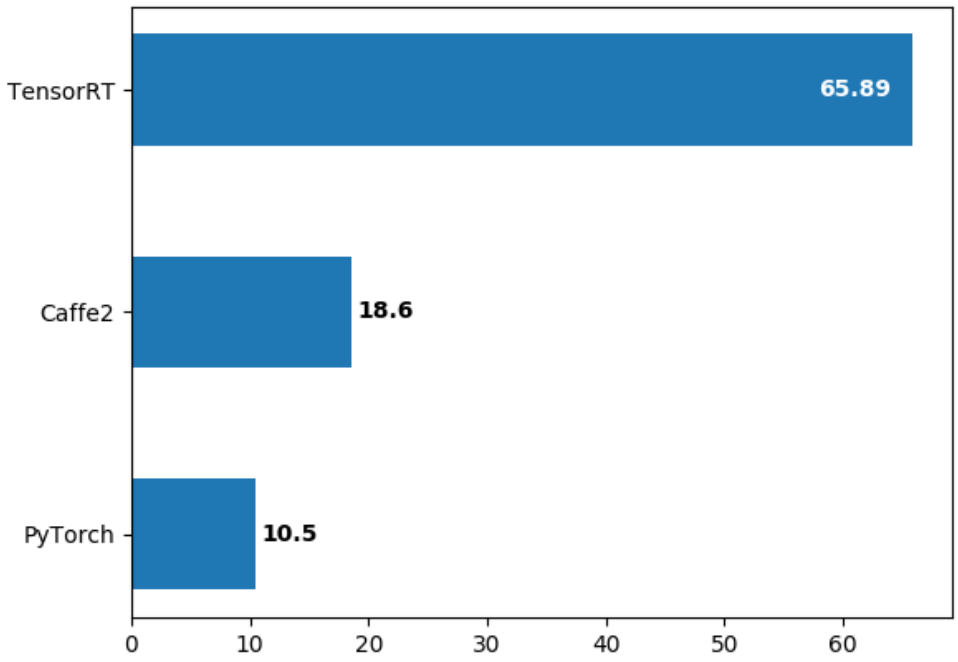



Figure 5.11: Runtime tests on the improved model for the three frameworks. The numbers on the bar are in units of FPS, and higher values are better.

estimation method that uses ITOP as the validation dataset. In the previous experiment on the ITOP dataset 5.5.1, the $mAP@10cm$ of our improved model was slightly higher than that of the state-of-the-art method VI [9] and the random tree walk (RTW) [49]. In terms of accuracy, our method does not comprehensively outperform VI and RTW because there are still some joints with lower accuracy than those of VI and RTW. However, in terms of the frame rate, VI takes 1.6 seconds to calculate each frame with 10 iterations, and RTW takes 0.1 seconds. [9] did not describe the equipment they used to test the runtime frame rate.

# Chapter 6

# Conclusions

## 6.1 Conclusions

In this thesis, we investigate the existing depth image datasets and CNN-based models for 3D human pose estimation. The existing datasets consist of everyday movements and do not contain rapid or large-scale movements. Therefore, the models trained from these datasets are not suitable for physical education. The existing models with depth maps as input can be divided into three categories: end-to-end models, heat map-based models, and hybrid models. End-to-end models perform direct regression from the depth image to

coordinates, making it difficult to reach high precision. Heat map-based models generate heat maps for each joint and then process them into coordinates, which is unfriendly to model deployment. Hybrid models use heat maps as intermediate features and perform regression to coordinates using $soft - argmax$.

Therefore, in this thesis, we first propose a 3D human pose estimation dataset called the Kivi dataset to address the lack of single-person large-scale movements in existing datasets. This dataset contains only the depth maps collected by Kinect, the joint coordinate values collected by the Vicon motion capture system, and the action number. Therefore, this dataset is only suitable for 3D human pose estimation and action recognition tasks with depth images as input.

We then design a preliminary model based on a state-of-the-art model to first estimate the xy-coordinates of joints and then extract the depth of the corresponding pixels from the depth image according to the xy-coordinates. However, due to occlusion, this method is only effective for joints that are not occluded. To solve the occlusion problem, we design an improved model. The improved model extracts the depth map as three heat maps for each key point. These three heat maps are projections of the 3D volumetric heat map on three faces. Then, the coordinates are computed by $soft - argmax$ from the heat maps. As the coordinates are calculated, the coordinates of each axis are obtained from the average of the calculation results of the two heat maps. Therefore, the coordinates of one axis can affect the branches of two heat maps at the same time during backpropagation. The preliminary model on the side view of the ITOP dataset reaches 63.67% mAP at 10 cm, whereas the improved model obtains 82.16%. The two models are trained and tested on the Kivi dataset. The preliminary model obtains 46.61% mAP at 10 cm, and the improved model obtains 53.95%. To verify that the improved model can be deployed in real time, we simulate the online environment and find that the model can work at 65.89 FPS with the TensorRT framework on an Nvidia 1080 Ti GPU.

## 6.2 Limitations and Future Work

This study has potential limitations.

First, the proposed dataset contains only depth images and annotations for the joint locations. In contrast to popular datasets for human pose estimation tasks, such as the ITOP dataset, our dataset does not have pixel-level segmentation for the background and foreground or point clouds. We can improve this in subsequent studies by manual labeling or by adopting a machine learning algorithm.

Second, the depth images in the Kivi dataset contain noise due to both the Vicon camera and Kinect emitting infrared light and capturing the reflected infrared. Some studies have made efforts to fill the hole-like noise on the depth images captured from the Kinect. Similar work can be classified as depth image denoising.

Finally, in contrast to 2D pose estimation for color images, the current study can only estimate the pose of a single person from the depth map. Addressing this problem involves building up a dataset containing multiple people in a single depth image, which is challenging in laboratory conditions. For tasks related to rehabilitation and physical training, multiperson pose estimation is meaningless. We suggest that the focus of research should be on 3D pose estimation from color pictures, and the problem of optical blur can be solved by building a new dataset with high-framerate cameras.

# References


[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[2] Long Chen et al. "Cross-View Tracking for Multi-Human 3D Pose Estimation at over 100 FPS". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3279–3288.

[3] François Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[4] Ben Crabbe et al. "Skeleton-free body pose estimation from depth images for movement analysis". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2015, pp. 70–78.

[5] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: 2005.

[6] A. El Saddik. "Digital Twins: The Convergence of Multimedia Technologies". In: *IEEE MultiMedia* 25.2 (2018), pp. 87–92. ISSN: 1070-986X. DOI: 10.1109/MMUL.2018.023121167.

[7] Dominik Maria Endres and Johannes E Schindelin. "A new metric for probability distributions". In: *IEEE Transactions on Information theory* 49.7 (2003), pp. 1858–1860.

[8] Michael Grieves. "Virtually Intelligent Product Systems: Digital and Physical Twins". In: July 2019, pp. 175–200. ISBN: 978-1624105647. DOI: 10.2514/5.9781624105654.0175.0200.

[9] Albert Haque et al. "Towards viewpoint invariant 3d human pose estimation". In: *European Conference on Computer Vision*. Springer. 2016, pp. 160–177.

[10] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[11] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[12] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[13] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

[14] Weiting Huang et al. “AWR: Adaptive Weighting Regression for 3D Hand Pose Estimation”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2020.

[15] “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems”. In: *IEEE Std. 1588-2008* (2008). DOI: 10.1109/IEEESTD.2008.4579760.

[16] 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* 36.7 (2013), pp. 1325–1339.

[17] Yangqing Jia et al. “Caffe: Convolutional architecture for fast feature embedding”. In: *Proceedings of the 22nd ACM international conference on Multimedia*. 2014, pp. 675–678.

[18] Gunnar Johansson. “Visual perception of biological motion and a model for its analysis”. In: *Perception & psychophysics* 14.2 (1973), pp. 201–211.

[19] Anthony Jones and Jim Ohlund. *Network Programming for Microsoft Windows*. Microsoft Press, 1999.

[20] Hanbyul Joo et al. “Panoptic Studio: A Massively Multiview System for Social Interaction Capture”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[21] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[23] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[24] Sijin Li and Antoni B Chan. "3d human pose estimation from monocular images with deep convolutional neural network". In: *Asian Conference on Computer Vision*. Springer. 2014, pp. 332–347.

[25] Min Lin, Qiang Chen, and Shuicheng Yan. "Network in network". In: *arXiv preprint arXiv:1312.4400* (2013).

[26] David G Lowe et al. "Object recognition from local scale-invariant features." In: *International conference on computer vision*. Vol. 99. 2. 1999, pp. 1150–1157.

[27] Diogo C Luvizon, Hedi Tabia, and David Picard. "Human pose regression by combining indirect part detection and contextual information". In: *Computers & Graphics* 85 (2019), pp. 15–22.

[28] Dushyant Mehta et al. "Monocular 3d human pose estimation in the wild using improved cnn supervision". In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 506–516.

[29] David L Mills. "Internet time synchronization: the network time protocol". In: *IEEE Transactions on communications* 39.10 (1991), pp. 1482–1493.

[30] David L Mills. "Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI". In: (2006).

[31] 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.

[32] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked hourglass networks for human pose estimation". In: *European conference on computer vision*. Springer. 2016, pp. 483–499.

[33] Aiden Nibali et al. "3d human pose estimation with 2d marginal heatmaps". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 1477–1485.

[34] Nvidia. *NVIDIA TensorRT: Programmable Inference Accelerator*. `https://developer.nvidia.com/tensorrt`. Accessed Feb 19.2020.

[35] Timo Ojala, Matti Pietikainen, and David Harwood. "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions". In: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 1. IEEE. 1994, pp. 582–585.

[36] Adam Paszke et al. "PyTorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems*. 2019, pp. 8024–8035.

[37] Michalis Raptis, Darko Kirovski, and Hugues Hoppe. "Real-time classification of dance gestures from skeleton animation". In: *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*. ACM. 2011, pp. 147–156.

[38] Romer Rosales and Stan Sclaroff. "Inferring body pose without tracking body parts". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. IEEE. 2000, pp. 721–727.

[39] Indrani Roy. *Why Digital Twin Is Trending Across Industries*. `https://aits.org/2018/05/why-digital-twin-is-trending-across-industries/`. Accessed Dec 30.2019.

[40] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[41] Siemens. *Digital Twins*. https://new.siemens.com/global/en/company/stories/research-technologies/digitaltwin/digital-twin.html. Accessed Dec 30.2019.

[42] Leonid Sigal and Michael J Black. "Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion". In: *Brown Univertsity TR* 120 (2006).

[43] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[44] Piotr Szczuko. "Deep neural networks for human pose estimation from a very low resolution depth image". In: *Multimedia Tools and Applications* (2019), pp. 1–21.

[45] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[46] Claire Thomas. *Built for speed: what makes Usain Bolt so fast?* https://www.telegraph.co.uk/usain-bolt-worlds-fastest-man/0/built-for-speed-what-makes-usain-bolt-so-fast/. Accessed Dec 31.2019.

[47] Paul Viola, Michael Jones, et al. "Rapid object detection using a boosted cascade of simple features". In: *Conference on computer vision and pattern recognition*. Vol. 1. 511-518. 2001, p. 3.

[48] Jingwei Xu et al. "Deep Kinematics Analysis for Monocular 3D Human Pose Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 899–908.

[49] 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.

[50] Zhengyou Zhang. "Microsoft kinect sensor and its effect". In: *IEEE multimedia* 19.2 (2012), pp. 4–10.