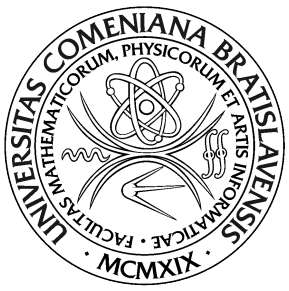
COMENIUS UNIVERSITY BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND
INFORMATICS

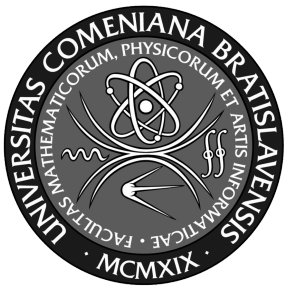


SKELETON TRACKING USING DEEP LEARNING

Master thesis

2019 Bc. Dana Škorvánková

COMENIUS UNIVERSITY BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND
INFORMATICS

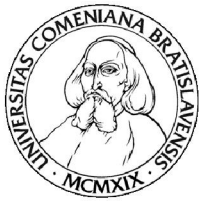


SKELETON TRACKING USING DEEP LEARNING

Master thesis

Program of Study:	Applied Informatics
Field of Study:	2511 Applied Informatics
Department:	Department of Applied Informatics
Supervisor:	RNDr. Martin Madaras, PhD.

Bratislava, 2019 Bc. Dana Škorvánková



Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Dana Škorvánková
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Skeleton Tracking using Deep Learning
Skeleton Tracking using Deep Learning

Anotácia: (C++, neural network)

Cieľ:
- skeleton tracking of a person
- input is a sequence of RGB or RGBD images
- use neural network to approximate the body movement

Vedúci: RNDr. Martin Madaras, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 25.09.2017

Dátum schválenia: 03.10.2018

prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

I hereby declare I wrote this thesis by myself, only with the help of referenced literature, under the careful supervision of my thesis supervisor.

.....

Bratislava, 2019 Bc. Dana Škorvánková

Acknowledgement

I would like to thank ...

Abstrakt

Abstrakt...

Kľúčové slová: odhad pozície kostry človeka, hlboké učenie, neurónové siete, mračná bodov

Abstract

Abstract...

Keywords: human pose estimation, deep learning, neural networks, point clouds

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deep learning was first introduced as a machine learning research area in 1986, with the aim of shifting the concept of machine learning closer towards artificial intelligence. During the last decade, artificial neural networks have become one of the most frequently used methods of machine learning in various research fields, such as computer vision, robotics, medicine, manufacturing, telecommunications, automotive engineering, and many more.

Neural networks are able to carry out numerous different tasks, which essentially consist of classification, prediction, clustering and associating. Regarding the classification task, the neural network learns to organize patterns or data into a number of predefined categories. Classification algorithms are often used to solve issues like medical diagnoses, e-mail spam filtering, speech recognition, handwriting recognition or image recognition. Concerning prediction (or regression) tasks, the aim is to produce the expected output from the given input data. Clustering is an unsupervised task, which concludes of a classification of input data based on an identification of a unique feature of the data, without any predefined classes. This technique is widely used

for pattern recognition, feature extraction, data mining, image segmentation etc. The associating task means the neural network is capable of storing or remembering certain patterns, thus it associates the previously unseen data with the most comparable pattern in its memory. This is mostly used in the field of pattern recognition and pattern completion.

One of many fields where the neural networks are applicable is human motion analysis. Among the most frequent motion tasks are skeleton tracking, human motion prediction and pose estimation. The motion tasks using either data-based methods, which rely on motion capture systems, or physics-based methods, which depend on optimization to predict motion, still remain a challenge these days.

In our thesis, we will be focusing on the task of pose estimation. The main goal of our study is to implement a method for 3D human pose estimation from depth input data using a deep learning approach. Our intention is to build a convolutional neural network to perform the prediction of 3D skeleton joint positions. In our work, we are going to implement several models and evaluate the results obtained on a number of benchmark datasets.

The thesis is organized in the following manner: In the next chapter of our work, we are going to illustrate basic motivation that lead us to devote ourselves to the stated issue, and indicate the current deficiencies in the problematics. In the third chapter, we will summarize an overview of neural networks used in image processing and take a look specifically on the 3D pose estimation task. The current state of research in this field and related work is discussed in the fourth chapter. The fifth chapter is dedicated to our proposed implementation of the given task, that is, the technologies we are going to use

for the implementation, the benchmark datasets on which we will evaluate our results, and the models we are going to implement. In the subsequent chapter, we will focus on the actual implementation of the proposed methods, we will go through implementation details and problems encountered during the process. The evaluated results of our thesis are reviewed in the seventh chapter. Finally, in the last chapter, we will sum up the goals of our thesis, the achieved results, and the conclusions of our work.

Chapter 2

Motivation

The task of human pose estimation attracts a lot of attention among deep learning researchers, mainly because of its frequent usage in virtual and augmented reality, action recognition, surveillance, human-robot interaction, trajectory prediction or motion-based human identification. Although a lot has been achieved in the 3D human pose estimation task, there are still many challenges nowadays, which are not easy to overcome.

Since most of the research is currently focused on estimating the pose from RGB data, one of the most critical challenges of pose estimation from 3D input is data availability. To successfully train a neural network of reasonable size, the large and well labeled dataset is crucial. Right now, there is a very small set of publicly available 3D human pose estimation databases. Moreover, even among the available datasets, it is hard to find one that is both large enough in its scale, and accurate enough to avoid overfitting of the neural model. There are several large action recognition datasets with motion capture ground truth, but since their primal purpose is not to provide the exact skeleton joint locations, the ground truth is often not accurate enough

for the task of pose estimation.

Due to the lack of the accessible depth data, many researchers have recently used their own recorded depth datasets to evaluate the results of their proposed method. However, this leads to the fact, that it is difficult to objectively compare the particular methods to each other, because the recorded databases are often not published. It is important to mention, that recording of a quality depth dataset is not a trivial task, mainly since the expensive motion capture system is usually required to obtain accurate ground truth labels, which also limits us to indoor scenes. The usual workaround is to use the Kinect camera for recording, which can also directly extract the 3D skeleton joint coordinates, even though still working well only in indoor scenes.

Another issue concerning pose estimation from 3D data is the actual type of 3D data that is passed as input to the neural network. The most frequent option is to use depth maps, thus encoding the third dimension into the 2D image. Anyhow, the depth maps are very dense representation of a 3D human pose. Also, a big part of the image space is representing the surrounding scene and background clutter, and the actual human body is only a small part of the scene. That results in expensive computations and lowering the time efficiency, while also processing the seemingly redundant data. On the other hand, in the attempt to overcome these drawbacks, voxelized grids have been used in several solutions to provide sparser 3D data representation. Voxelized grids are basically point clouds, discretized to the predefined grid values. However, voxels have their shortcomings too. One thing is that voxels require 3D convolution operations, which are rather demanding in terms of memory, time, and computing power. Moreover, the conversion of point clouds into the voxelized grids can be time-consuming itself.

The aim of our thesis is to develop a 3D human pose estimation pipeline based on neural network, which takes 3D data (in a form of a point cloud or a depth map) as input and outputs the 3D skeleton joint coordinates. Our goal is to propose our own model, in addition to implementing several well-performing models proposed in existing papers. Next, we aim to evaluate them on benchmark datasets, and compare the results with the current state-of-the-art.

Chapter 3

Overview

In this chapter, we will go through general overview of the techniques we are going to use in our solution. We will define basic terms associated with neural networks and their functioning, illustrate several scenarios on how neural networks can be utilized in image processing, and explain the concept of the pose estimation task, along with the various sub-categories we distinguish among.

3.1 Neural networks

In general, artificial neural networks were introduced as structures inspired by the biological structure of human brain. Similarly to the brain, the basic computational units of the artificial neural networks are neurons, which are connected together by synapses. In neural networks, synapses are simply weight values, which means the neuron performs some kind of calculation and the result is multiplied by the value corresponding to the particular connection it is passing through. The neurons are organized in layers, which is the main idea of the so-called deep learning. The neural networks can

perform both supervised and unsupervised learning.

In most cases, we are dealing with the supervised learning, where we feed the model with the input data and also provide the output we expect the network to produce. As the input data is passed through the network in what we call a forward-pass, the neurons are outputting certain computed values and passing them to the neurons they are connected to, which in the end form a single or multiple outputs of the network. The idea of the supervised learning is that at the end of the forward-pass, we provide a feedback to the network, about how correctly it performed the desired task. This is done by comparing the actual output of the model to the desired output. The comparison is also referred to as calculating a loss function. The calculated loss is then back-propagated through all the layers in the network to adjust the weights of the model accordingly. This scenario is repeated in many iterations, while the loss value computed on the model output should be decreasing.

Nowadays, as we have enough computational power and the amount of available data large enough to train on, the capacities and architectures of the neural networks are becoming increasingly extensive and powerful.

3.1.1 Image processing using deep learning

Neural networks have shown their significant contribution to various research fields, and one of them is digital image processing. Many different tasks concerning the image processing can be carried out by a neural network, e.g. image classification, feature extraction, pattern recognition, object detection, image captioning etc. For instance, in image classification and feature extraction, deep learning has shown many advantages over the traditional

methods, which rely on handcrafted features. Using a neural network model, instead of manually selecting and extracting image features, we can pass the image directly and the model learns to find and distinguish the important features itself, and additionally, might classify the image accordingly. While the first layers often recognize very simple features, like edges and corners, the deeper we dive into the network architecture, the more complex features are sought in particular layers and neurons within them.

3.1.2 3D pose estimation

One of the tasks concerning image processing and human motion analysis, frequently performed by neural networks, is the 3D pose estimation. As the name implies, it is a task of estimating the three-dimensional pose of a human subject from a single image or set of video frames. The resulting pose is determined by 3D coordinates of the skeletal joints of the human body. The number of joints can vary, thus it is usually considered a hyperparameter. The more skeletal joints (and output coordinates) the model estimates, the more complex representation of the human body skeleton we obtain.

We mentioned the neural model infers the pose from an image, however, the actual form of the input data may vary. First, the individual approaches to the task can be divided in two categories according to the input data dimension. Two-dimensional input data are usually RGB images. The important advantage of such approaches is that they can be easily utilized in real-time in-the-wild applications, since a common RGB camera is sufficient to capture the images. This is also a reason, why a majority of the pose estimation research nowadays is focused on inferring from 2D data, as well as most of the public pose estimation datasets consist only of RGB images.

On the other hand, three-dimensional input data bring an additional depth information into the neural networks, which is often very handful in increasing the accuracy of the estimation. The three-dimensional input data comes in various forms. The most common are depth maps. The basic idea behind depth maps is to encode the third dimension into the two-dimensional image, where at each pixel location, the corresponding pixel value is representing the value of the third coordinate (or depth) at the specified position. Other than depth maps, a convenient way to capture three-dimensional data is to use point clouds. Anyhow, point clouds have several attributes which are not suitable for neural networks. Most importantly, point clouds are unordered and irregular. Some of the researchers resolved this by making use of the voxelized grids, that is, discretized the point clouds into a predefined grid. Alternatively, several approaches evolved on the idea of processing the unorganized point clouds using shared multi-layered perceptrons to obtain the features of the point cloud and work further with them.

Another classification of the pose estimation methods is into model-based generative methods and discriminative methods. ...

Chapter 4

Related work

Nowadays, neural networks are widely used in the field of image processing, pattern recognition, human movement analysis and many more. There are numerous types of tasks concerning human movement analysis, where the neural networks proved to be beneficial, e.g. action recognition, action classification, body-movement-based human identification, pose estimation etc.

Focusing on the pose estimation task, there have been many different methods and approaches presented in recent years. Based on the type of the input data, the studies can be divided into approaches inferring from two-dimensional data (RGB images) and three-dimensional data (depth maps, point clouds, voxelized grids etc.). The two-dimensional approaches are far more usable and easily accessible in real-time applications, being able to run without any special devices, using only the RGB camera. On the other hand, the regression of 3D joint positions from 2D input data requires highly non-linear operations, what can lead to many difficulties in the learning procedure. The three-dimensional approaches provide the additional depth in-

formation, which can significantly simplify the task for the network, and thus improve the estimation accuracy.

4.1 Two-dimensional input data

Among studies working with the RGB data, Mehta et al. [8] introduced a system called *VNect* to obtain real-time full global 3D skeletal pose, combining a pose regressor based on convolutional neural network with kinematic skeleton fitting. They also invented a new concept of the so-called *location maps*, that are produced by the network, along with the confidence heatmaps for each joint. The network outputs three location maps and one heatmap per joint. The predicted 3D position of each body joint is read out from three location maps, corresponding to x, y, and z coordinate, at the estimated pixel location of the particular keypoint, determined by the heatmap. The pipeline introduced in the stated paper is shown in Fig. 4.1.

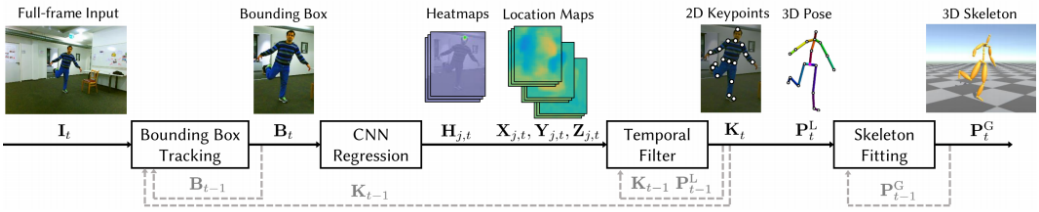


Figure 4.1: The overview of the VNect pipeline [8].

However, the stated model was unable to handle occlusions or capture multiple people in the scene. Thus, they removed these restrictions in the follow-up model called *XNect* [9], which is able to capture multiple people in the scene by a single RGB camera. Unlike the previous work, the model outputs full skeletal pose in joint angles and global body positions of a coherent skele-

ton in real-time. The method consists of three subsequent stages: First, a convolutional neural network estimates 2D and 3D features and identity assignments for all visible body joints of all subjects in the scene, second, a fully-connected neural network turns possibly occluded 2D and 3D pose features into a complete 3D pose per subject, and third, a space-time skeletal model fitting is applied to the predicted 2D and 3D pose to merge them and enforce temporal coherence.

In another paper, Mehta et al. [7] focused on RGB data as well, but this time, they presented a method based on the concept of transfer learning. They embraced the similarity between the feature extraction tasks in 2D and 3D space, and transferred the features learned on 2D data to 3D pose estimation network (that is, pre-trained the 3D convolutional model on 2D features). In the paper, they put a great emphasis on *in-the-wild* scenes, that is, the pipeline was designed to perform well even outside of the recording room, in generic outdoor scenes. Newell [11] presented stacked hourglass networks, consisting of multiple stacked hourglass modules which allow for repeated bottom-up, top-down inference. These networks are designed such that the features are processed across all scales, allowing them to capture the spatial relationships within the body. Chou et al. [2] employed generative adversarial networks as the learning paradigm. They consist of two stacked hourglass networks – generator and discriminator, where the former is used for the human pose estimation and the latter back-propagates the adversarial loss between the ground-truth and the generated output to the generator.

Rogez et al. [15] decided to utilize the concept of the pre-defined anchor poses. Their proposed pipeline consists of three sub-tasks. First, they extract candidate regions using a Region Proposal Network [14] and place the

set of anchor poses into the predicted bounding boxes. Then, they run it through a classification branch, which outputs probabilities of anchor poses to be correct at each location. Finally, the regression branch computes an anchor-pose-specific regression that estimates the difference between the pose proposal and ground-truth pose, and refines the final pose estimation.

4.2 Three-dimensional input data

The three-dimensional data used as the input to the neural networks come in various forms. Most frequently, 3D input data are in a form of depth maps (RGB-D images). Depth maps are actually encoding 3D space into 2D image, where the value at each pixel position represents the corresponding depth value (third axis coordinate). Marin-Jimenez et al. [6] proposed a technique where the final estimated pose is computed as the weighted sum of the predefined set of prototype poses. The weights corresponding to the prototypes are directly regressed from input depth maps by a convolutional neural network. In the paper, they claim to have outperformed state-of-the-art results on the benchmark datasets and reached 100% accuracy in mean average precision at 10cm on the ITOP dataset [3], i.e. every skeleton joint is predicted within 10cm range from its ground-truth position. The stated approach is an example of a single-stage method.

The two-stage methods generally consists of the segmentation stage and the regression stage. First, the input data is segmented to the corresponding body-parts. Then, the segmented input data is used to infer 3D joint coordinates. An example of a two-stage method was proposed by Shafaei and Little [16]. They treat the problem of 3D pose estimation from depth data through two-stage pipeline, where in the first stage the body parts are identified in

the input depth maps by a dense classifier. In the second stage, all camera views are merged, and the created unified 3D point cloud is passed to a linear regressor to compute 3D body joint locations.

Aside from depth maps, some of the methods make use of the voxelized grids, made by discretizing a given point cloud in a predefined set of values. However, voxels require use of three-dimensional convolutions, what makes working with them very time-consuming and computationally expensive. *V2V PoseNet* [10] operates with this kind of data and regresses joint locations with 3D CNN-autoencoders. They first use 3D CNN encoder and decoder to estimate per-voxel likelihood of each skeleton joint from voxelized input (Fig. 4.2). Afterwards, they refine the target object localization with a 2D CNN which takes a cropped depth map and output an offset from the its reference point to the center of ground truth joint positions. This way, they obtain an accurate reference point.

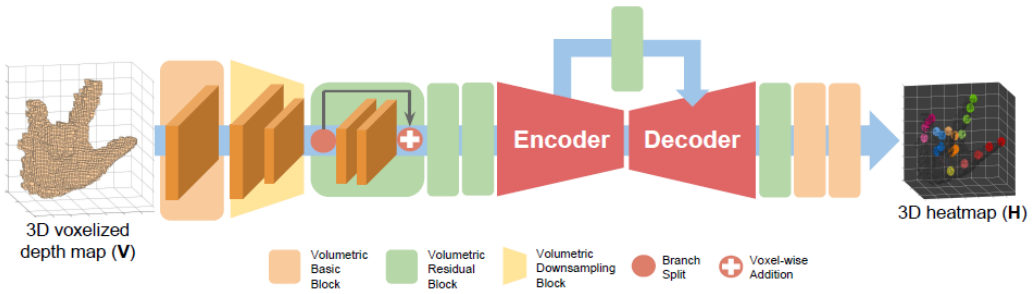


Figure 4.2: The V2V-PoseNet architecture which takes voxelized input and estimates the per-voxel likelihood for each keypoint [10].

Nevertheless, there are several networks proposed which work with unordered point clouds as input data, yet implement the convolution operations on the point clouds without using computationally expensive 3D convolutions. Some

of the methods decided to use shared multi-layered perceptrons and max-pooling layers to obtain the features of point cloud. Although they extract global features, since the max-pooling layers are applied on the whole set of points, it is hard to capture local features.

Qi et al. [12] proposed a classification and segmentation model called *PointNet*, where they intended to incorporate the local features by aggregation of the intermediate outputs from the classification network, before and after max-pooling. Afterwards, they fed the aggregated local and global features into the segmentation network. Later, Qi et al. [13] introduced *PointNet++* model, which has similar key structure as the previous PointNet, but improves the model by utilizing a hierarchical structure, similar to the one used in image convolutional neural networks. It recursively applies PointNet on a nested partitioning of input point cloud, starting from small local regions and gradually extending to bigger regions. The whole hierarchical feature learning architecture (on 2D point set example) suggested in the paper is shown in Fig. 4.3.

In another study, Wu et al. [18] presented a new convolution operation called *PointConv*, which can be applied on unordered and irregular point clouds. They treat convolution kernels as nonlinear weight and density functions of the local coordinates of 3D points. The weight functions are learned with multi-layer perceptron networks and density functions through kernel density estimation. Such learned kernels can be used for translation-invariant and permutation-invariant convolutions on any 3D point set.

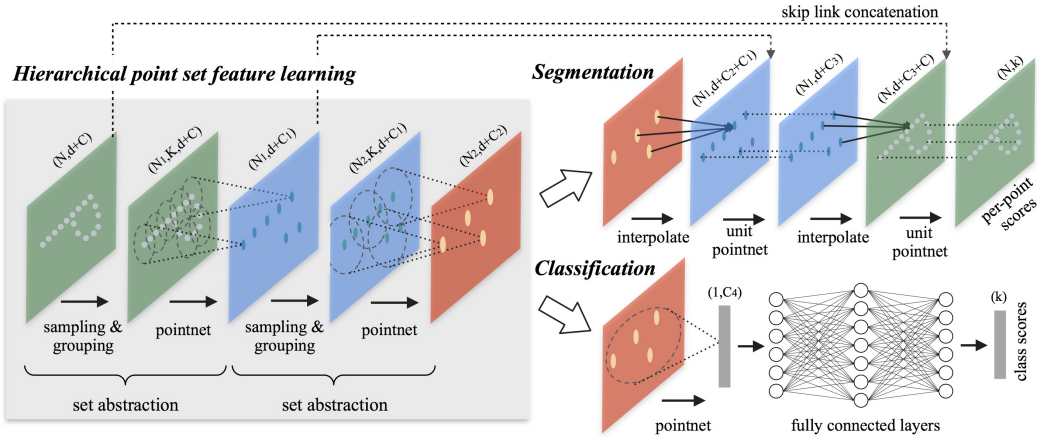


Figure 4.3: The PointNet++ hierarchical feature learning architecture shown on points from 2D Euclidean space [13].

It is worth mentioning, that all of the stated methods processing unordered point clouds perform object classification or segmentation task, which is not an aim of our thesis. Concerning pose estimation task, Ali [1] introduced a novel one-stage approach in his thesis called *Point-Based Pose Estimation* (PBPE), taking point clouds directly as input data to the model which outputs 3D skeleton joint coordinates. He concludes, that since point clouds are able to provide sparser representation of the human body than depth maps, the operations on them would be much easier, and thus, the computational complexity would be reduced. The inspiration for the model was in the PointNet architecture. Besides the proposed PBPE model, the contribution of his work also consists of the refinement of several two-stage methods by using an automatic annotation mechanism for labeling body regions in real data. Next, the study presents the benefits of fusion of the real training data and more complex synthetic training data. The poses in the synthetic dataset are much more varied, so by adding certain amount of the synthetic data to the real dataset during the training phase, they extend the diversity

of the training set. As a result, the model is able to generalize better. On the other hand, the synthetic data is also useful for pre-training a model, reducing the computational cost and time of the real data annotation. Thus, such pre-trained model can be fine-tuned on relatively small part of the real dataset, yet achieving reasonable results.

Chapter 5

Proposed implementation

In this chapter, we are going to describe the implementation we propose, the technologies we intend to use and the models we aim to examine.

5.1 Technologies

In this thesis, we implement all of the studied models in python deep learning library Keras with Tensorflow backend. All of the experiments are conducted on NVIDIA GTX 1070.

5.2 Benchmark datasets

In this section, we are going to describe the datasets which will be used for the evaluation purposes of this thesis. All of the datasets mentioned below were made publicly available for research purposes.

5.2.1 ITOP

The Invariant-Top View Dataset (ITOP) [3] consists of approximately 50k real-world depth images from two camera viewpoints (front view and top view). It captures 20 people, each performing 15 different actions. The dataset comes with the initial partition of the data into the train and test set. The train and test data contain around 18k and 5k samples from each viewpoint, respectively. Besides the depth images and the ground-truth joint labels, the dataset also includes raw point clouds. The skeletal model in this dataset is described by 15 body joints. Sample depth maps from the dataset are shown in Fig. 5.1. As can be seen in the figure, the depth images are rather noisy, but the noise can be partly reduced by several background segmentation methods.

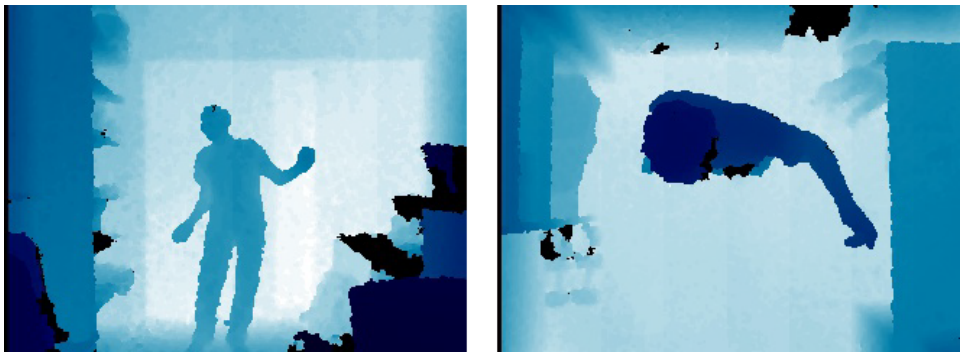


Figure 5.1: Sample depth images from the ITOP dataset [3] (front and top view).

The current state-of-the-art on ITOP dataset was claimed in [6], where the 100% accuracy at 10cm precision was reached, which means every joint was predicted within 10cm from its ground-truth position. In the paper, they also claim to have reached the mean error of approximately 0.9cm on ITOP dataset (0.19cm on the front-view data), which is the best result achieved on

the stated dataset, up to our knowledge.

5.2.2 UBC3V

The UBC3V [16] is a synthetic dataset made for the task of pose estimation from multiple cameras. It contains around 6 million synthetic depth frames structured in three parts according to the complexity of the human postures – easy, medium and hard pose. The pose in each frame is represented by the position of 18 skeletal joints. It captures a total of 16 characters and each frame is observed from three different viewpoints. Therefore, the samples can be treated both as multi-view (point clouds from three cameras merged into one), or single-view postures (each sample handled separately). Although the multi-view point clouds describe a particular posture in a more complex way, considering the use-case in most of the real-time applications, the single-view samples are often the preferred option, due to the difficulty of the task of multiple camera synchronization.

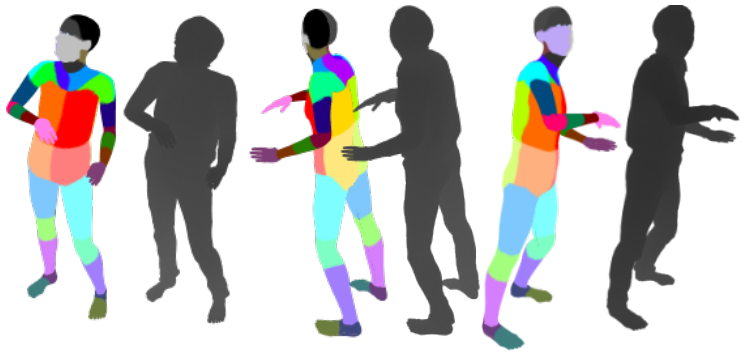


Figure 5.2: Sample data from the UBC3V dataset (the figure shows the same pose from three different cameras) [16].

The depth data are available in the form of depth images, but can be converted into point clouds in world reference coordinates using the intrinsic

and extrinsic camera parameters. The ground truth labels are composed of 18 joints per posture, and the dataset also comprises a segmentation of each point cloud into 43 body regions (as can be seen in Fig 5.2).

5.2.3 MHAD

The Berkeley Multimodal Human Action Database (MHAD) [17] was recorded on real humans. It contains 11 actions performed by 7 male and 5 female subjects. Each subject performed each of the actions 5 times, which yields about 660 action sequences corresponding to about 82 minutes of total recording time. The total number of depth frames is over 250k. The skeleton structure in this dataset is defined by 35 joint locations. Two of the performed actions involve a chair, used for the subject to sit down and stand up. It is worth a mention, that the chair itself provides a lot of clutter in the depth data. The whole dataset was captured by multiple devices, including cameras, depth sensors, accelerometers and microphones. Two kinect cameras were used and synchronized to acquire the depth data, one placed in front of the subject, the other one at the back. Intrinsic and extrinsic camera parameters are provided to extract the scene as a point cloud with real-world coordinates.

5.2.4 CMU Panoptic Dataset

[4]

5.3 Approaches

The goal of our study is to implement several different approaches to the task of human pose estimation using neural networks. Among the methods we implement are both existing state-of-the-art neural models, as well as our

novel approach we propose in this thesis – a two-stage pipeline called Four-channel Pose Estimation. We present the existing models we re-implement, and our novel approach in the subsequent sections.

5.3.1 Deep Depth Pose model

One of the aims of this work, is to implement the Deep Depth Pose model (DDP) [6] in Keras framework. We consider this an essential step to propose a state-of-the-art pose estimation model performing on depth data, since the DDP model claims to outperform all of the present methods on the examined datasets. Originally, the model is implemented in Matlab and while some parts of the code (mostly testing) are publicly available, the training procedures have not been published.

The main idea behind the stated method is incorporating a set of predefined prototype human poses. The convolutional neural network outputs a vector containing weights, each corresponding to particular prototype pose. The weighted prototypes are then summed up producing the final estimated pose. The predefined prototype poses should ideally cover the largest possible range of human poses, so that by their linear combination we can obtain any desired pose during the testing. Hence, the set of prototype poses is formed by applying K-means clustering on the training dataset.

5.3.2 Point-Based Pose Estimation model

The Point-Based Pose Estimation model (PBPE) [1] is an approach to 3D human pose estimation from depth data, remarkable by several of its attributes. The whole network consists of two branches, the regression branch and the auxiliary segmentation branch. The output of the model is a 3D body pose,

given by the coordinates of skeletal joints. One of the benefits of the method is the possibility of omitting the auxiliary branch in the testing phase, where the model is already trained, and by this restriction of the model's depth and complexity, the computational cost and prediction time can be cut down.

Of great significance for our thesis is the fact, that the input to the PBPE network is directly in a form of an unstructured point cloud. This may be convenient for a number of reasons, one of them being the acquisition of a sparser representation of the human body pose, in comparison to commonly used depth maps. The more density the input covers, the more computations have to be performed in the network, therefore point clouds offer more effective data processing even in complex model architectures. The model architecture is inspired by PointNet model [12]. Inside the PointNet as well as the PBPE model, the point clouds are processed using the pseudo-convolutions with kernel size 1×1, which are generally employed to change the filter space dimensionality, mostly reducing the number of depth channels.

Since the source code for the PBPE model has not been made public prior to working on this study, we aim to implement the model in Keras framework from scratch and evaluate the results on benchmark datasets.

5.3.3 Four-channel Pose Estimation

An important part of our work is to develop a novel method for human pose estimation. We introduce the Four-channel Pose Estimation – a two-stage pipeline, which takes a point cloud as an input, and outputs the 3D coordinates of the estimated skeletal joint positions. Retaining the idea of handling point clouds directly with pseudo-convolutions, first stage of the pipeline is based on a segmentation network, which classifies the points representing a

human pose into the corresponding body regions. In the second stage, the input point cloud containing the point coordinates is concatenated with the output regions from the segmentation network, thus forming a four-channel point cloud input. Such produced data, conserving together the local as well as the global information, is then fed into the second model – the regression network, where the joint coordinates are finally regressed.

Similarly to the models mentioned above, we implement our novel approach in Keras framework as well. To be able to compare the performance of our model to that of the existing ones, the same benchmark datasets are used for the evaluation.

Chapter 6

Implementation

This chapter is dedicated to the description of the implementation details of the particular methods and neural models. We will introduce the architecture of the models, as well as the data pre-processing, training procedures and optimization parameters used.

6.1 Deep Depth Pose model

In this section, we are going to describe our implementation of the DDP model (proposed in [6]) in Keras framework. The follow-up additional modifications of the model are presented at the end of the section.

The DDP model is based on the idea of linearly combining the predefined prototype poses to obtain the resulting pose estimation. The set of prototype poses is produced by clustering the training dataset into K clusters, K being a hyperparameter of the network. The output of the model represents K weights, each corresponding to one of the prototype poses. The final estimated pose is then obtained as a sum of the weighted prototypes.

6.1.1 Model structure

The architecure of the DDP model itself is relatively simple. The model is actually not very deep, it consists of five convolutional blocks, each followed by ReLU activation. First three blocks also contain a pooling layer, as indicated in Fig 6.1. After the convolutional blocks, there are three fully-connected layers. The number of neurons in the output layer is a hyperparameter set to the number of prototype poses clustered from the dataset. The input to the model is a one-channel depth map of size 100 x 100 pixels.

Input	Conv01	Conv02	Conv03	Conv04	Conv05	Full01	Full02	Full03
100×100	$7 \times 7 \times 96$ P: 2×2	$5 \times 5 \times 192$ P: 2×2	$3 \times 3 \times 512$ P: 2×2	$2 \times 2 \times 1024$	$2 \times 2 \times 2048$	1024 Dr=0.2	256	K

Figure 6.1: Deep Depth Pose model architecture [6] (P stands for pooling layer, Dr indicates dropout, K is the number of prototype poses).

6.1.2 Loss function

The loss function used during training the model is defined as huber loss with a regularization term, as described in Eq. 6.1, where $\mathcal{L}_R$ is the (residual) huber loss.

$$\mathcal{L}_{DDP}(g(\mathbf{D},\theta),\mathbf{C},\mathbf{p},\alpha) = (1-\alpha)\cdot\mathcal{L}_R(\mathbf{C}\times g(\mathbf{D},\theta),\mathbf{p}) + \alpha\cdot\|g(\mathbf{D},\theta)\|_1 \quad (6.1)$$

In the equation above, $g(\mathbf{D},\theta)$ represents a non-linear function computed by the neural model on the input depth map $\mathbf{D}$ with trainable parameters θ, returning a column vector of length K (number of clustered prototype

poses), $\mathbf{C}$ is a matrix with K columns containing prototype poses, $\mathbf{p}$ stands for the vectorized ground-truth pose, and α is a regularization coefficient – a hyperparameter to control the magnitude of the resulting weights of the prototypes.

6.1.3 Initialization

The values of the model's hyperparameters from the original DDP paper were chosen using the 'Mann-Whitney U-test' [5], thus determining, whether the median difference between pairs of configurations are statistically significant. The value of the regularization coefficient was fixed to $\alpha = 0.08$ for the ITOP dataset, and $\alpha = 0.01$ for the UBC3V dataset. The number of prototypes was set to $K = 70$ and $K = 100$ regarding the ITOP and the UBC3V dataset respectively.

The model learns using the Adam optimizer with the learning rate set to 10^{-3}, which is progressively decreasing during the training. The weights in the model are initialized using the Xavier normal initializer, that is randomly chosen from the normal distribution with zero mean and a standard deviation proportional to the filter size. The biases are initially set to zero. The size of batches is set to $b = 64$.

6.1.4 Data pre-processing

In the pre-processing stage, the depth maps are re-sized to match the model input dimensions. First, the image is cropped along the larger dimension on both sides to fit the square, assuming the subject is located near the center of the image. Then, the image is re-sized to 100×100 px. Afterwards, the depth values are normalized to range [0,1]. Also, the input depth images are

normalized globally by subtracting the mean image of the training set from each depth sample. The ground-truth poses are being normalized as well – to zero mean and one standard deviation.

Concerning the datasets used for evaluation, UBC3V already comes with the split into the train, validation and test set. The ITOP dataset is originally divided into the train and test set only, thus the validation set was acquired by random sampling of the train data. The ratio was set to 80/20, meaning the data used for validation are representing 20% of the train set. During the experiments, only the hard-pose part of UBC3V, and the side-view part (for single-view approach) of the ITOP dataset were used.

6.1.5 Modifications

This section introduces several modifications we made to the proposed DDP model, in order to lower the mean error of the predictions. Since the experience of the researchers made us believe that the implementation of the same model in different framework often leads to different results, we diverted from the original architecture of the model, as well as the initial optimization parameters.

However, still inspired by the DDP model, we have built a slightly deeper, more complex model, in an attempt to improve the final estimations. We used both ITOP and UBC3V datasets again, for the evaluation of the modified model. The architecture of the modified DDP network is shown in Fig. 6.2. As shown in the figure, we significantly increased the amount of dropout layers, as the model tended to overfit the training data. Moreover, two additional convolutional layers were stacked in the model architecture.

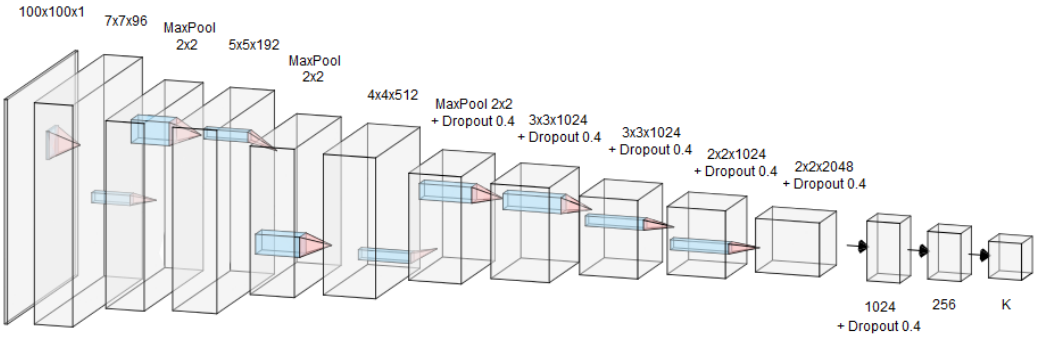


Figure 6.2: Our modification of the original DDP model architecture (K stands for the number of prototype poses).

Besides the structure of the model, some of the hyperparameters were slightly modified as well, mainly due to switching to another deep learning framework, where the back-end procedures differ, which may be resulting in a convergence to different outputs. The regularization parameter was fixed to $\alpha = 0.1$ for both benchmark datasets. The residual huber loss was replaced by an absolute error, and the L2 norm was used in the regularization term instead of the L1 norm. Also, the initial learning rate was reduced to 10^{-4}.

Regarding the test phase, the experiments on the ITOP dataset have shown that even though the test set seems sufficiently representative (that is, retains the approximate distribution of the training set), the error on the test data is by far worse than the error on previously unseen data obtained by random sampling of the training set (used as a validation set). This is probably caused by the fact, that the input data is formed by a sequence of video frames, thus many subsequent samples captures almost the same content, with only a minimal shift between the frames. As a result, a lot of the validation samples are nearly the same as the ones used for training.

The final results of both the original DDP and modified DDP model, evaluated on the test set of ITOP and UBC3V datasets, are reported and discussed in Chapter 7.

6.2 Point-Based Pose Estimation model

As another part of our study, we chose to examine the Point-Base Pose Estimation (PBPE) model [1]. Again, we re-implement the method in Keras framework, though the original implementation was in pure Tensorflow. However, note that the original code was not published prior to our implementation.

The model is heavily inspired by the implementation of the PointNet [12], which was built for the task of object classification and semantic segmentation. Basically, the architecture has been modified to fit the task of human pose estimation, making use of the auxiliary sub-network contributing to the global model loss.

6.2.1 Model structure

The architecture of the proposed model consists of two branches (or sub-networks). The auxiliary sub-network is included to compute the body part segmentation on the fly, while the main network regresses the joint locations. The basic idea behind the structure of the model is the aggregation of both local and global features of the input point cloud in the auxiliary part-segmentation network. Without the sub-network, almost all of the local context would be lost because of the max-pooling operation in the intermediate layers. The incorporation of the local features helps the network

understand the relationships among particular local regions of the human body. The whole architecture of the stated model is shown in Fig. 6.3.

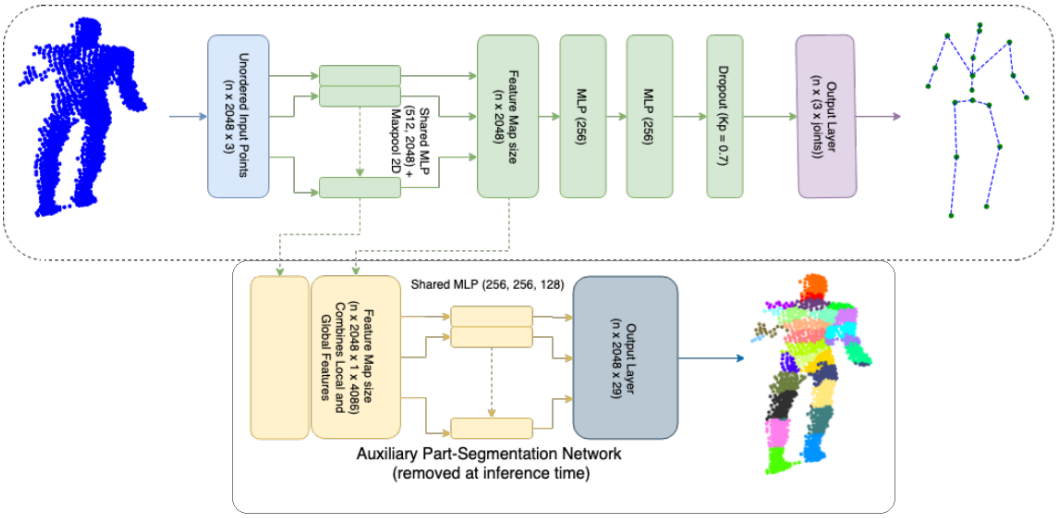


Figure 6.3: The Point-Based Pose Estimation model architecture [1].

As already mentioned, the unorganized and irregular depth input represented by a point cloud is a key factor in this architecture. For this very reason, the engagement of the classic convolution operation would be meaningless, since the point cloud has provides no explicit spatial information about the neighboring points. In this case, pseudo-convolutions with a kernel size of 1×1 are employed. The function of such pseudo-convolution layers is reducing (or expanding) the dimensionality in the filter space. A stack of the 1×1 convolution layers, each followed by ReLU activation, is illustrated in Fig. 6.3 as a shared multilayer perceptron (shared MLP). The numbers in the parentheses indicate the number of filters in particular layers.

The input of the model is of shape $(b \times p \times 3)$, where b represents the batch size, and p is the number of points in the input point cloud, each located by

three coordinates. The output shape of both sub-networks depends on the number of joints in the skeleton structure. The number of joints, as well as the number of points the point clouds are containing, is a hyper-parameter.

The input passed to the auxiliary segmentation sub-network is formed by concatenating the local and the global features obtained in the regression branch. The local features contains the outputs of pseudo-convolutions before the max-pooling is applied. The global features are acquired as the result of max-pooling across all points in the point cloud. The auxiliary sub-network outputs per-point labels segmenting the input point cloud into particular body parts.

One of the conveniences of the PBPE model is the optional omission of the auxiliary branch at test time. Since we are focusing strictly on the task of pose estimation, the body part segmentation is only relevant for us in a context of training (and contributing to the model loss). Hence, we can prevent the data from being passed through the segmentation branch at the inference time, saving computational cost and time.

6.2.2 Loss function

As a loss function of the regression branch, the model uses a simple mean squared error, which in our implementation has been replaced by mean absolute error, as it helped the model to converge faster in our case. Regarding the segmentation branch, the categorical cross-entropy is being used, since it is essentially a classfication task.

According to our experiments, without the segmentation sub-network, the model tends to slightly overfit the train data. The loss of the auxiliary

segmentation branch is contributing to the global loss to help the model generalize better on previously unseen data. Originally, the weight of the contribution was set to $w_a = 0.1$. The weight of the global loss was initially set to $w_g = 0.9$, however given by our observations, we fine-tuned the parameter to $w_g = 1$, and decreased the weight of the segmentation branch to $w_a = 0.01$. Again, the differences between the performance with the particular hyper-parameter values might be caused by the change of the deep learning framework.

6.2.3 Initialization

The hyper-parameters of the model mostly preserve the same values as in the original paper, except for the weight initialization, the initial learning rate and loss weights mentioned in the previous section. The weights inside the model are sampled from the normal distribution with zero mean and standard deviation relative to filter size, using the Xavier initializer. The initial learning rate is set to 10^{-3}, and is reduced exponentially with a decay rate $d = 0.2$ in each epoch. The model is trained using the Adam optimizer. In each step, the model processes a batch of size $b = 32$.

Another essential hyper-parameter is the predefined number of points in the input point clouds. The value is set to $p = 2048$, therefore every point cloud needs to be sub-sampled to this number of points, prior to feeding it to the network. The number of joints in the skeleton representation, defining the output shape of the both model branches, should be adjusted according to the dataset the network is processing at the moment.

6.2.4 Data pre-processing

Depending on the single or multi-view approach, the input point clouds are treated each as a single sample, or the different viewpoints of the same frame are merged into a single point cloud. Based on the existing studies, inference from multi-view input data is expected to give more accurate results, however the single-view based estimation is far more useful in most of the real-time applications, since there is no need for camera synchronization. Therefore, we have carried out experiments inferring from multi-view, as well as single-view input data.

As the next pre-processing step, the point clouds were sub-sampled to the specified number of points ($p = 2048$) using the farthest point sampling technique. The input point clouds and the corresponding joint locations were normalized to the range $[-1, 1]$ using the minimum and maximum values of the whole training set. Concerning the validation and test data, there were two options, how to normalize them – either using the scaling parameters of the training set, or scale the validation and test set to the specified range with its own parameters. However, since the model is designed to work in a scenario, where the test data are fed into the network one-by-one, the only possible option is scaling it using the known parameters of the train set.

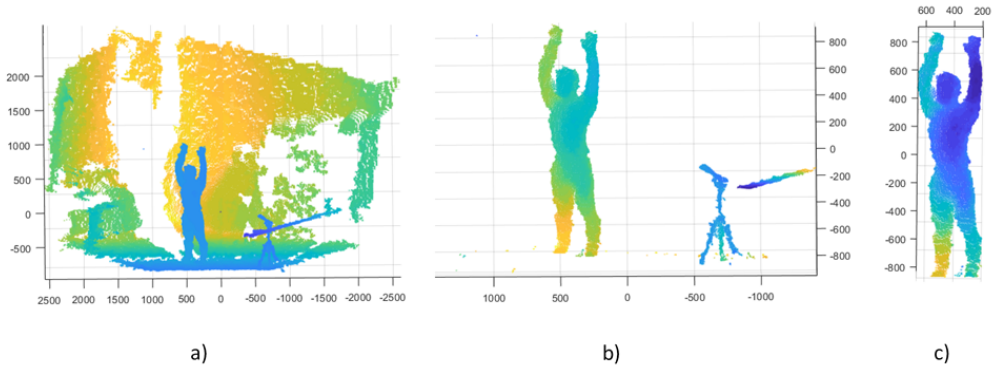


Figure 6.4: Data pre-processing shown on a sample point cloud from MHAD dataset [17] a) before segmentation, b) after removing the background and floor, c) after final clustering.

Since the used datasets (except for the UBC3V) captures real-world data, the raw point clouds contain a lot of noise. Therefore, we consider suitable pre-processing as a very important step. First, the background wall and ground floor are removed by MSAC plane fitting (which is a variation of RANSAC algorithm). Then, the segmentation of the point cloud into clusters based on Euclidean distance is performed, and the biggest cluster is considered as the desired subject. Fig. 6.4 depicts a sample from MHAD dataset, before and after applying the segmentation pipeline.

The MHAD dataset does not provide any partition into the train and test data. Thus, the test data was selected as randomly sampled 25% of the whole dataset during the experiments, while the rest was marked as the train set. As already pointed out in the original proposal of the PBPE model, the skeleton representation in the MHAD dataset is very detailed, yielding a total of 35 skeletal joints. Though this provides the ability of estimating the pose with higher complexity, some of the joints have the locations in the skeleton representation so close to each other, they may be considered

redundant when estimating a full-body pose. Among these joints are two pairs of nodes located at toe tips and one additional pair of nodes representing the fingertips. TODO compare 29 vs 35 joints in Table - in Results chapter As shown in Table ..., it is interesting to observe how the number of joints in the skeleton (and the complexity of the model) influences the pose estimation results.

As the real datasets do not comprise the partition into the particular body regions, the automated technique for the annotation has been proposed in the original implementation of PBPE model. Each skeletal joint is assigned one body region. Using the automatic annotation, each point of the point cloud is associated with the closest skeletal joint based on Euclidean distance. The example of the annotated point cloud is shown in Fig. 6.5.

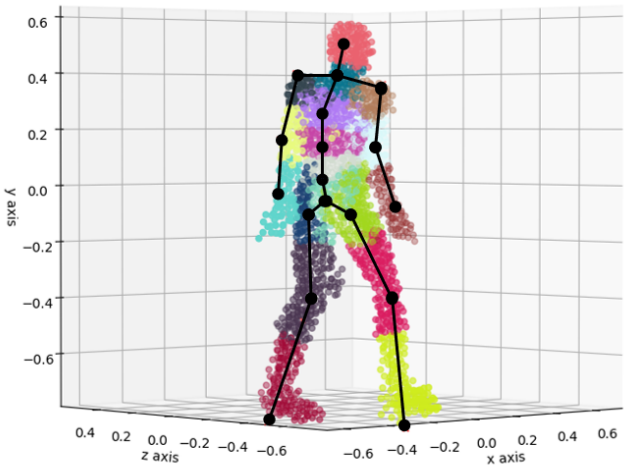


Figure 6.5: The automatic body region annotation on a sample real data point cloud.

6.3 Four-channel Pose Estimation

This section is dedicated to the implementation of a novel approach we present as a part of the thesis contribution. The core of the proposed method is a two-stage pipeline, consisting of the body segmentation network and the regression network. The two subsequent stages take a point cloud as an input and produce skeletal joint coordinates as a result.

6.3.1 Model structure

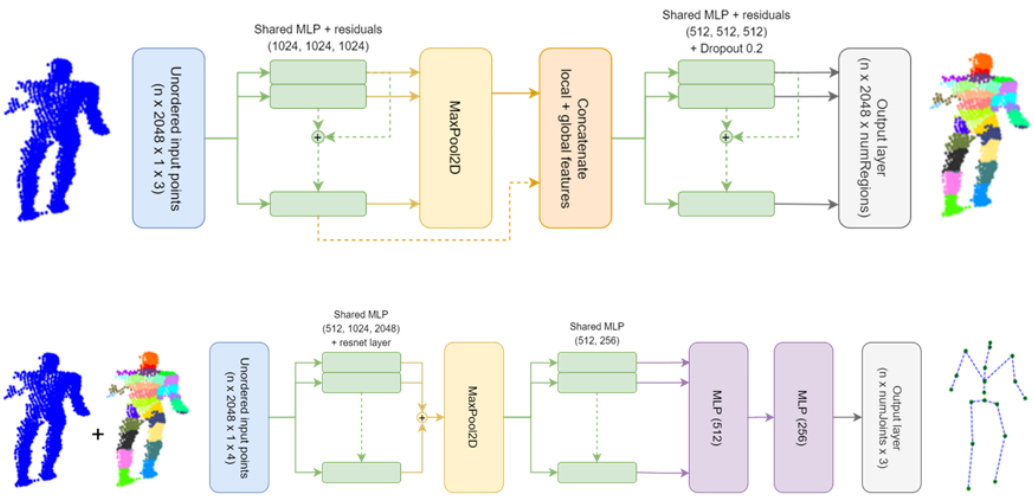


Figure 6.6: The Four-channel Pose Estimation pipeline: First, point clouds are segmented into body regions in the segmentation network (up), then the input point clouds are concatenated with the predicted body regions as a fourth channel, and are fed into the regression network (down).

6.3.2 Loss function

6.3.3 Initialization

Chapter 7

Results

Chapter 8

Conclusion

Bibliography

[1] Ali, Ashar. 2019 (May). *3D Human Pose Estimation.* MS Thesis, Georgia Institute of Technology.

[2] Chou, Chia-Jung, Chien, Jui-Ting, & Chen, Hwann-Tzong. 2017. Self Adversarial Training for Human Pose Estimation. *CoRR*, **abs/1707.02439**.

[3] Haque, Albert, Peng, Boya, Luo, Zelun, Alahi, Alexandre, Yeung, Serena, & Fei-Fei, Li. 2016 (October). Towards Viewpoint Invariant 3D Human Pose Estimation. *In: European Conference on Computer Vision (ECCV).*

[4] Joo, Hanbyul, Simon, Tomas, Li, Xulong, Liu, Hao, Tan, Lei, Gui, Lin, Banerjee, Sean, Godisart, Timothy Scott, Nabbe, Bart, Matthews, Iain, Kanade, Takeo, Nobuhara, Shohei, & Sheikh, Yaser. 2017. Panoptic Studio: A Massively Multiview System for Social Interaction Capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[5] Mann, H. B., & Whitney, D. R. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Ann. Math. Statist.*, **18**(1), 50–60.

[6] Marin-Jimenez, M.J., Romero-Ramirez, F.J., Muñoz Salinas, R., & Medina-Carnincer, R. 2018. 3D Pose Estimation from Depth Maps us-

ing a Deep combination of Poses. *Journal of Visual Communication and Image Representation*. In press.

[7] Mehta, Dushyant, Rhodin, Helge, Casas, Dan, Fua, Pascal, Sotnychenko, Oleksandr, Xu, Weipeng, & Theobalt, Christian. 2017a. Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision. *In: 3D Vision (3DV), 2017 Fifth International Conference on*. IEEE.

[8] Mehta, Dushyant, Sridhar, Srinath, Sotnychenko, Oleksandr, Rhodin, Helge, Shafiei, Mohammad, Seidel, Hans-Peter, Xu, Weipeng, Casas, Dan, & Theobalt, Christian. 2017b (July). VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. vol. 36.

[9] Mehta, Dushyant, Sotnychenko, Oleksandr, Mueller, Franziska, Xu, Weipeng, Elgharib, Mohamed, Fua, Pascal, Seidel, Hans-Peter, Rhodin, Helge, Pons-Moll, Gerard, & Theobalt, Christian. 2019. XNect: Real-time Multi-person 3D Human Pose Estimation with a Single RGB Camera. *CoRR*, **abs/1907.00837**.

[10] Moon, Gyeongsik, Chang, Ju Yong, & Lee, Kyoung Mu. 2017. V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation from a Single Depth Map. *CoRR*, **abs/1711.07399**.

[11] Newell, Alejandro, Yang, Kaiyu, & Deng, Jia. 2016. Stacked Hourglass Networks for Human Pose Estimation. *CoRR*, **abs/1603.06937**.

[12] Qi, Charles Ruizhongtai, Su, Hao, Mo, Kaichun, & Guibas, Leonidas J. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CoRR*, **abs/1612.00593**.

[13] Qi, Charles Ruizhongtai, Yi, Li, Su, Hao, & Guibas, Leonidas J. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *CoRR*, **abs/1706.02413**.

[14] Ren, Shaoqing, He, Kaiming, Girshick, Ross, & Sun, Jian. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Pages 91–99 of:* Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., & Garnett, R. (eds), *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc.

[15] Rogez, Grégory, Weinzaepfel, Philippe, & Schmid, Cordelia. 2019. LCR-Net++: Multi-person 2D and 3D Pose Detection in Natural Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 01.

[16] Shafaei, Alireza, & Little, James J. 2016. Real-Time Human Motion Capture with Multiple Depth Cameras. *In: Proceedings of the 13th Conference on Computer and Robot Vision*. Canadian Image Processing and Pattern Recognition Society (CIPPRS).

[17] Vidal, Rene, Bajcsy, Ruzena, Ofli, Ferda, Chaudhry, Rizwan, & Kurillo, Gregorij. 2013. Berkeley MHAD: A Comprehensive Multimodal Human Action Database. *Pages 53–60 of: Proceedings of the 2013 IEEE Workshop on Applications of Computer Vision (WACV)*. WACV '13. Washington, DC, USA: IEEE Computer Society.

[18] Wu, Wenxuan, Qi, Zhongang, & Li, Fuxin. 2018. PointConv: Deep Convolutional Networks on 3D Point Clouds. *CoRR*, **abs/1811.07246**.