國立臺灣大學電機資訊學院電子工程學研究所

碩士論文

Graduate Institute of Electronics Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

基於點雲及局部關節網路之人體姿態及體型估測

3D Human Pose and Shape Estimation from Point Clouds with Local Joint Network

章孟治

Meng-Chih Chang

指導教授：簡韶逸 博士

Advisor: Shao-Yi Chien, Ph.D.

中華民國 110 年 02 月

February 2021

doi:10.6342/NTU202100293

國立臺灣大學碩士學位論文
口試委員會審定書

基於點雲及局部關節網路之人體姿態及體型估測

3D Human Pose and Shape Estimation from Point Clouds with Local Joint Network

本論文係章孟治君（R07943130）在國立臺灣大學電子工程學研究所完成之碩士學位論文，於民國110年01月20日承下列考試委員審查通過及口試及格，特此證明

口試委員：簡韶逸（指導教授）

施吉昌　陳銘宇

許志強

所長

致謝

到這邊，我聽著 King Crimson 的《Epitaph》邊寫著，碩班一路走來真的蜿蜒曲折，從一開始的題目六自由度物體姿態估測，中途換成三維空間人體關節預測，最後是以人體姿態與體型估測作為結尾；在沒有人帶領的情況下獨自面對這個對實驗室來說是全新的題目，每天看工作站盯著進度條等結果，是好是壞無從得知，只能一試再試，今天開始跑，明天可以看結果，成功了可以大笑，但我總是害怕明日會讓我絕望的哭泣，然而事實也是如此。

雖然在主要研究道路上孤獨，但其他地方還是受到許多人幫助，洋彬在最後我獨自一人時給了許多提醒跟建議，禹呈跟均澤一直以來都有提供諮詢，曼妤、宏璋和翊恣則是在面試和其他地方有所幫助，致緯時常提供如何做研究的資訊，柏煒給的是情緒上的窗口，華揚幫我弄了一些套件讓我能用，聖竣來拍了資料集跟幫我跑實驗，實驗室其他人提供各種協助，在新竹和其他各地的老友會聽我苦中作樂的嬉鬧，最後是老師願意讓我這樣收尾。

這份論文對我來說一直都是未完成品，我不知道怎麼完成，繼續下去的話什麼時候會完成，所以就終止在這邊吧，將來有什麼時間，有更多資源的時候，但願我會再想起那些未完成的部分。

2021.02.03 章孟治

doi:10.6342/NTU202100293

中文摘要

三維空間的姿態與體型估測是屬於電腦視覺裡進階的題目，其最終目標為得到人體模型。相較於傳統的方法仰賴模型貼合法，現今的作法利用捲積神經網路來抽取深層特徵來取得人體模型的參數。現今頂尖的方法使用單張彩色影像當作輸入，然而由於渲染器裡的柵格化，三維空間中的幾何特徵是被隱含在其中。我們認為這樣的資料無法正確的帶出三維空間的資訊。因此我們認為使用具有三維空間資訊的資料，也就是深度影像或點雲，會是一個更好的選擇。

在這篇論文裡，我們提出一個兩階段的方法，透過深度影像或對應的點雲預測三維空間人體姿態與體型。這其中我們設計了兩個特殊的模組，都稱為局部關節網路。在第一個階段裡，我們先進行三維空間人體關節預測。我們假設事先可取得二維空間人體關節點來當作初始特徵，我們把這些關節點投影回三維空間形成初始三維空間關節，然後以這些關節為中心去對點雲進行分群，經過分群後的點雲會送進第一個局部關節網路來取的真正的相機空間中三維空間人體關節。在第二個階段裡，我們以前一階段得到的三維空間關節為初始特徵，結合點雲來預測人體模型參數，並使用另外一個局部關節網路來對這些參數進行細部修正。取得參數後，我們變能將它轉換成人體模型。我們驗證我們的方法在我們自己產生的合成資料上，其結果顯示我我們的方法是有效的。而我們論文的最終目標為實作一個擴增實境系統，為此，我們設計了一套系統，結合我們設計的模型，將 Kinect v2 相機的資料當作輸入，輸出具有擴增實境效果的圖。其結果也顯示我們的方法在實際資料上也是有效的。

doi:10.6342/NTU202100293

3D Human Pose and Shape Estimation from Point Clouds with Local Joint Network

Meng-Chih Chang

Advisor: Shao-Yi Chien

Graduate Institute of Electronics Engineering

National Taiwan University

Taipei, Taiwan

February 2021

doi:10.6342/NTU202100293

Abstract

3D human pose and shape estimation is an advanced problem in the Computer Vision region. The goal is to retrieve human meshes. While traditional methods mostly rely on model fitting method, modern approaches exploit the potential of Convolutional Neural Network (CNN) to extract deep features and regress the parametric representation of human meshes. Those state-of-the-arts use only an RGB image as input. However, because of the rasterization, the geometric features of 3D space are encoded implicitly in this kind of data. We argue that an RGB image cannot correctly bring out the information of 3D space. Therefore, we suggest that a 3D data, a depth image, or point clouds, would be a better choice.

In this thesis, we proposed a two stage method to predict human meshes from depth images or point clouds with two special modules named Local Joint Network (LJN). In the first stage, we predict 3D human joints first. We assume that the 2D joints are provided as initial information. We project those initial joints to 3D space by the depth and use the grouping technique to gather points into clusters according to them. The groups of features are sent to first LJN to predict the real 3D joints in camera coordinate. For the second stage, the 3D joints from the previous step will become initial features. We regress an initial parametric model according to point clouds and the initial features and then refine detailed parameters with another LJN. With the refined parameters, we can recover human meshes. We evaluate our method on a synthetic dataset generated on our own and the experiments show that our two models are effective. The final goal of our study is to achieve an AR system. To this end, we design a flow combining our models' outputs augmented

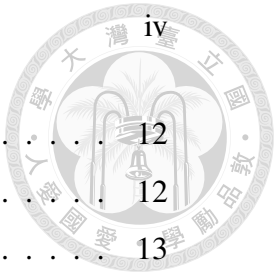
doi:10.6342/NTU202100293

images from Kinect v2 camera. The result also shows that our models work well even on real images.

doi:10.6342/NTU202100293

Contents

doi:10.6342/NTU202100293



doi:10.6342/NTU202100293

doi:10.6342/NTU202100293

List of Figures

doi:10.6342/NTU202100293

doi:10.6342/NTU202100293

doi:10.6342/NTU202100293

List of Tables

doi:10.6342/NTU202100293

Chapter 1

Introduction

Analyzing 3D data is a frequently discussed topic in computer vision. By understanding the 3D space, many applications, such as automatic manipulation, augmented reality (AR), virtual reality (VR), human-robot interaction, and so on, can progress. Particularly, human-related works drag lots of attention recently because of the commercial possibilities. In this thesis, we discuss two human-related problems: 3D human joint estimation and 3D human pose and shape estimation. Furthermore, we propose an AR system that utilize the contents of these two problems.

1.1 3D Human Joint Estimation

3D human joint estimation, also known as 3D human pose estimation, aims to locate the joints' coordinates in 3D space. Traditional methods range from geometry-based techniques [8] to purely statistical approaches [9]. Recent works rely on the powerful deep convolutional neural network (CNN) [10, 4, 11]. The two popular pipelines are let a network learns the mapping between 2D joints and 3D joints and direct prediction from RGB images. The latter one usually predicts 2D joints first as a side product too. However, these methods are easily confused by the lacking of depth. Some others use multi-view camera to deal with the problem. But the

doi:10.6342/NTU202100293

related camera position needs to be known.

In our method, we choose depth images as our input. Depth images encode 3D structure explicitly. Furthermore, we back-project the depth to the camera space with camera parameters to get the point clouds, which is a natural 3D data type. We propose a "grouping" operation that groups points into clusters and handle each cluster separably. The 2D joints will first be back-projected to 3D space, forming initial 3D joints. After that, we use the initial 3D joints as the centers to gather nearby points into clusters. The model takes clusters of points as input named Local Joint Network (LJN$_{jd}$, jd for joint detection). LJN$_{jd}$ consists of several sub-networks, and each of them can explore the local structure of a joint. Compared with direct prediction, our LJN$_{jd}$ will not be interfered by other joints' structures.

1.2 3D Human Pose and Shape Estimation

3D human pose and shape estimation, or human mesh recovery, intend to find a particular target's parametric model[5]. Early works [12] rely on model fitting, and the results are usually undesirable. Recent works [13, 14, 15] count on the deep features extracted by CNN from a single RGB image. Similarly, an RGB image cannot resolve the inherent ambiguity of 3D-2D projection.

To this end, we also choose depth images as our input. Besides, we propose another Local Joint Network (LJN$_{pse}$, pse for pose and shape estimation). In LJN$_{pse}$, we predict the parametric parameters hierarchically. Starting from the root joint, we refine the pose parameter sequentially according to the human kinematic tree. After the pose parameter is done, we move on to the shape parameter. The LJN$_{pse}$ will repeat several times to get a more acceptable result.

doi:10.6342/NTU202100293

1.3 System

A popular application of human pose and shape estimation is augmented reality. By knowing the target's pose, we can project the same mesh as the target or other meshes with the same pose on the display system. Therefore, we propose a system that combines the two models, 3D joint detection and 3D human pose and shape estimation, and outputs augmented images with human meshes from Kinect v2 camera. To accomplish the system, we train a special version of Cascaded Pyramid Network (CPN) [16] that predicts SMPL joints. With the 2D joints and the depth images, the rest is evident.

1.4 Contribution

To summarize our thesis, the main contributions are as follow:

- We estimate 3D human joint locations from depth images and 2D joints with Local Joint Network (LJN_{jd}), which explores the local structure of each joint.
- We estimate the parametric models (3D human meshes) from depth images and 3D joints with Local Joint Network (LJN_{pse}), which gets more refined results of each parameter.
- We introduce a system that achieves augmented reality with Kinect v2 camera.

doi:10.6342/NTU202100293

Chapter 2

Related Work

2.1 2D Joint Detection

2D joint detection, i.e., 2D human pose estimation, is a classical problem in computer vision. Generally, the problem can be classified into a key-points detection problem. Traditional[17, 18, 1] methods utilize pictorial structures[19, 20] that express the human body as a tree-structured graphical model based on human kinematic to solve the problem.

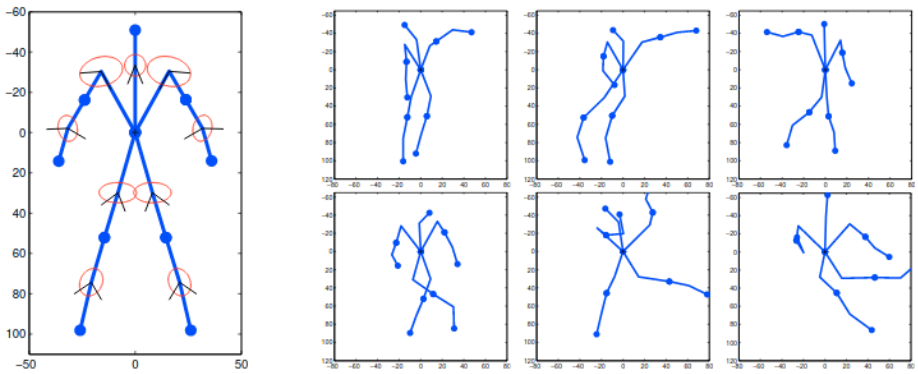
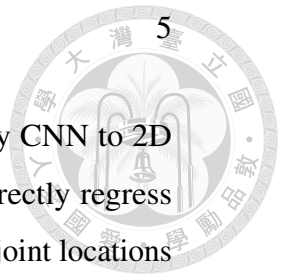


Figure 2.1: Pictorial structure example of [1].

As the convolutional neural network(CNN)[21, 22] rising, recent methods choose CNN as it's feature extractor to learn human joints' features and get

doi:10.6342/NTU202100293



remarkable results. DeepPose[23] is one of the pioneers that apply CNN to 2D human pose estimation. They use AlexNet[22] as backbone to directly regress joint locations. Despite the success of DeepPose, it's hard to predict joint locations directly. The probability map of the joint location, or heat-map, is considered a better representation. Newell et al.[2] use stacked hourglass modules to predict heat-maps of joints. Tompson et al.[24] and Wei et al.[3] utilize multi-resolution network and output heat-maps in different scales to predict joints coarse-to-fine.

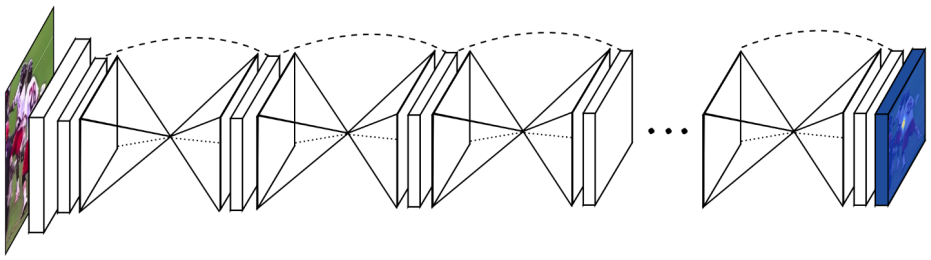


Figure 2.2: Stacked hourglass architecture of [2].

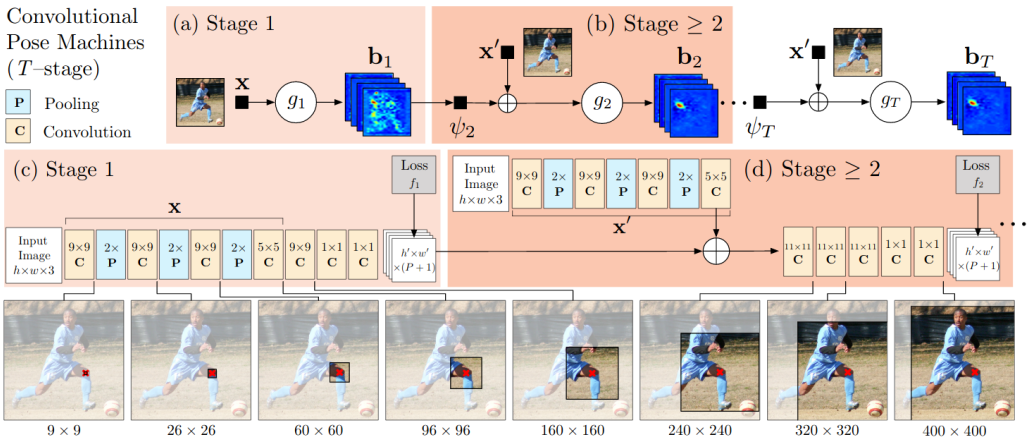


Figure 2.3: Mutil-scale network of CPM[3].

Different from single person 2D joint detection, multi-person joint detection needs to aggregate key points into several persons. Methods for multi-person tasks usually can be classified into two ways: top-down and bottom-up. Bottom-up approaches predict all key-points first and then combine joints that belong to

doi:10.6342/NTU202100293

the same person. DeepCut series[25, 26] assemble joints of a single person by partitioning predicted joints and labeled body parts. OpenPose[27] introduce part affinity fields(PAFs), which predict the possible connection between any two joints, to gather joints belong to the same person.

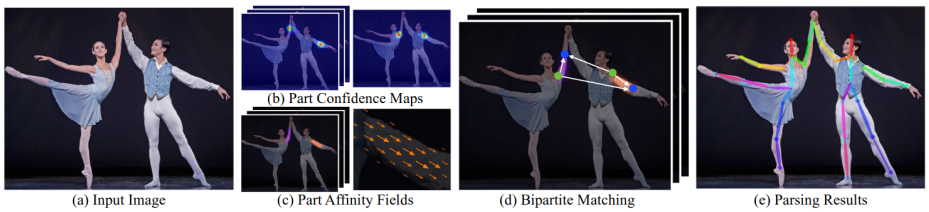


Figure 2.4: Pipeline of OpenPose.

In contrast to bottom-up, top-down methods[28, 29, 16] detect and crop out every person first and treat them as a single person case. Therefore, the performance is effected by both single person joint estimation and human detection.

2.2 3D Joint Detection

Due to the ambiguity between 2D and 3D space, 3D joint detection is a more advanced topic. Traditional methods[9, 8] require lots of additional priors. Recent methods exploit the power of CNN and tend to start from 2D joint to predict 3D joint, either assume hands-on 2D joint detection or train an end-to-end network by themselves. Moreno-Noguer et al.[10] transform 2D joints into Euclidean Distance Matrix(EDM) first and then regress another EDM of corresponding 3D joints. By Muldtidimensional Scaling (MDS), they can easily transform EDM back to 3D joints. Martinez et al.[4] use simple linear layer along with ReLU[30], batch normalization[31] and dropout[32] to achieve extraordinary result at that time, forming a simple baseline for 3D joint detection. Wu et al.[33] predict depth maps additionally, and let the network learn how to project 2D joints back to 3D joints in camera coordinate. Cheng et al.[11] take time information into consideration. Furthermore, they follow [34], use Kinematic Chain Space (KCS) in both spatial

doi:10.6342/NTU202100293

and temporal. Iskakov et al.[35] and Qiu et al.[36] exploit multi-view geometry, fusing features from different view by epipolar geometry and triangulation.

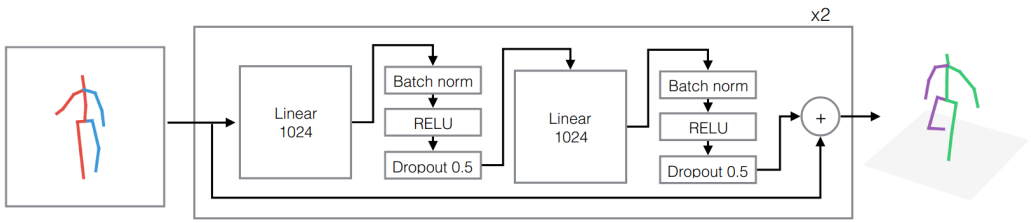


Figure 2.5: Architecture of [4].

2.3 Human Pose and Shape Estimation

3D human mesh recovery is also a classic problem in computer vision. Unlike typical mesh retrieval, human meshes are articulated objects. The different poses of one same person result in different meshes. Therefore, parametric human models[5, 37] are proposed to represent various poses and shapes.

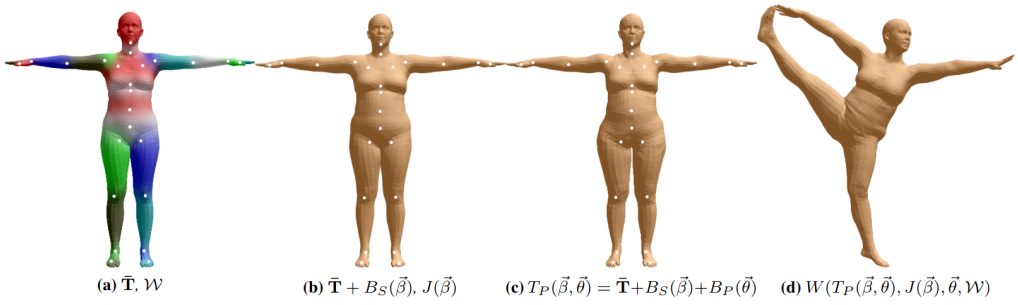


Figure 2.6: SMPL model [5].

2.3.1 From RGB Image

Bogo et al. [12] predict 2D joints first, and then fit them with projected 3D joints of SMPL model[5]. HMR[13] use iterative regression network to extract SMPL

doi:10.6342/NTU202100293

parameters and weak-perspective camera parameters from a single image. They also use a discriminator to verify valid human pose. Based on HMR, SPIN[14] concatenate the fitting procedure of [12] after it, HKMR[15] propose hierarchical estimation of pose parameters. Ci et al.[38] and Kolotouros et al.[14] utilize Graph Convolutional Network(GCN) to exploit structure information.

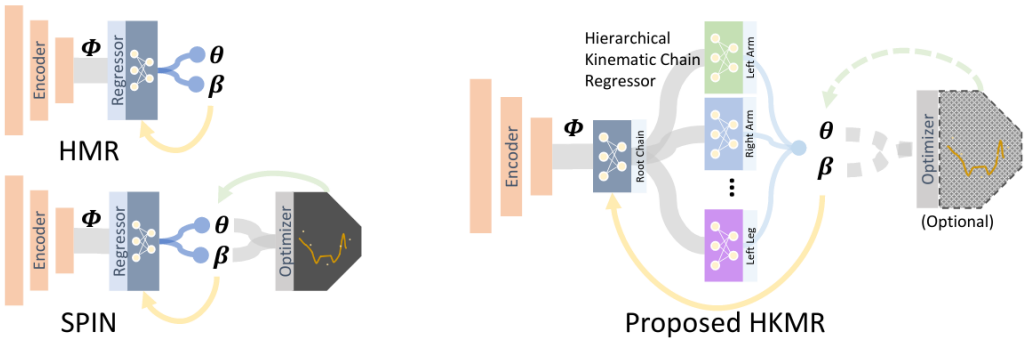


Figure 2.7: HMR, SPIN, and HKMR.

2.3.2 From Depth Image

Guo et al.[39] fit a pre-scanned template model into depth by L_0-based constraint, successfully reduce the deformation around joints. Fusion series[40, 41, 42] reconstruct human model by joint motion and surface motion. Wei et al.[43] take two depth maps as input, and use CNN to find the correspondences of them to construct mesh. Wang et al.[6] convert depth to point cloud first, and then exploit Pointnet[44, 45] and GCN to estimate vertex coordinate of SMPL mesh.

doi:10.6342/NTU202100293

Chapter 3

3D Human Pose and Shape Estimation from Point Clouds with Local Joint Network

As mentioned in the previous chapter, we estimate human meshes starting by detecting 3D joints first with LJN$_{jd}$. Afterward, we use the joints as initial features to predict SMPL parameters with LJN$_{pse}$. In the remainder of this chapter, we will first discuss how to represent a human mesh and the backbone we use, and then the detailed design of our models.

3.1 3D Human Body Model

Skinned Multi-Person Linear Model (SMPL)[5] is a linear and differentiable parametric model that factor human bodies into two parameters, pose $\theta \in \mathbb{R}^{3K}$ and shape $\beta \in \mathbb{R}^{10}$. The pose parameter represents the relative 3D rotation of K = 24 joints (the order is show in Fig.3.1) in axis-angle representation, affecting the deformation of a mesh's surface. The shape parameter is calculated as the first ten coefficients of a PCA projection of the shape space, defining individuals' varying

doi:10.6342/NTU202100293

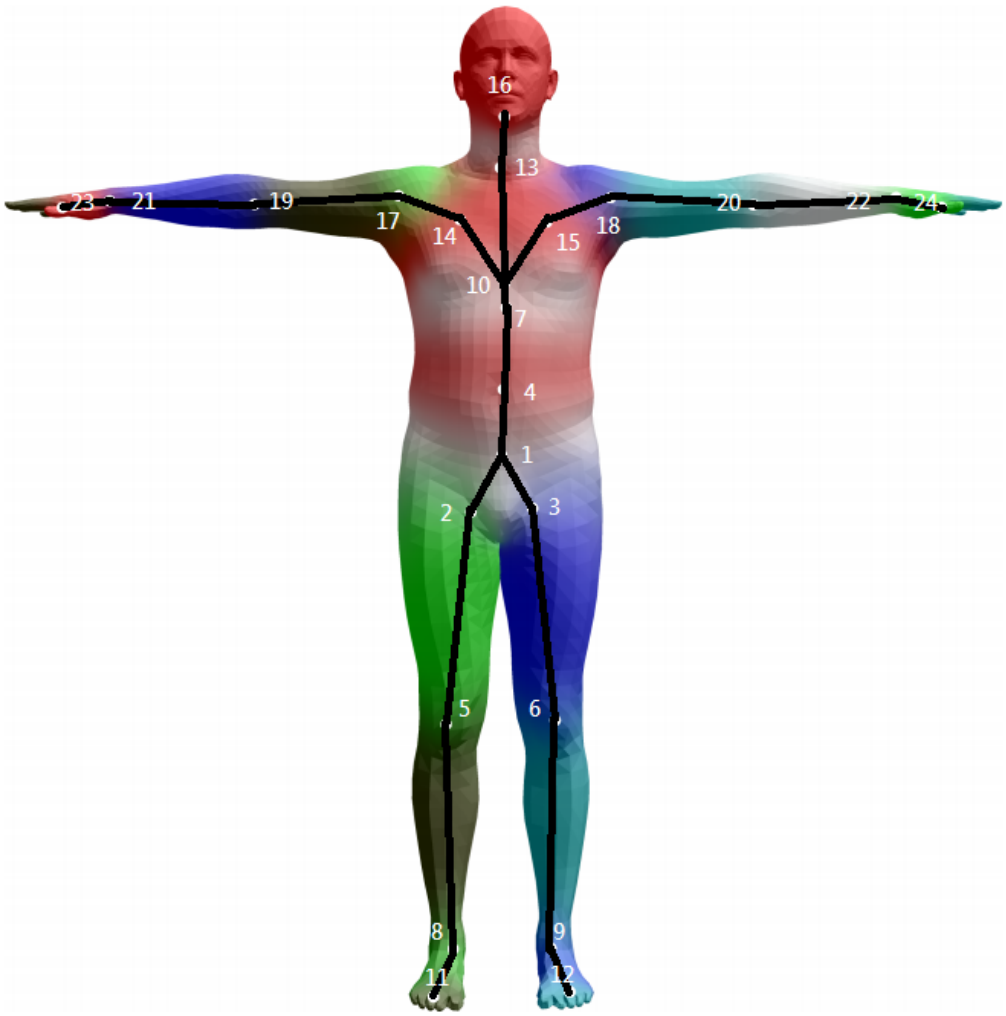


Figure 3.1: Numbering of SMPL joints. The figure are originally shown in [5].

physique. The whole transform of SMPL is differentiable and defined as follow:

$$M(\vec{\beta}, \vec{\theta}) = W(T_p(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W}) \tag{3.1}$$

$$T_p(\vec{\beta}, \vec{\theta}) = \overline{\mathbf{T}} + B_s(\vec{\beta}) + B_p(\vec{\theta}) \tag{3.2}$$

where $M(\vec{\beta}, \vec{\theta}) \in \mathbb{R}^{\mathcal{V}\times 3}, \mathcal{V} = 6980$ is the vertices coordinate of a transformed mesh, $\mathcal{W}$ is the blend weights, $\overline{\mathbf{T}}$ represents vertices of rest pose, and $B_s(\vec{\beta})$ and $B_p(\vec{\theta})$ describe the displacement of vertices from the rest pose to the target pose.

doi:10.6342/NTU202100293

3.2 Backbone Network

Our input data is a single depth map. To avoid heavy calculation, we project the depth map back to the camera space to get a set of 3D point clouds $\{P_i \in \mathbb{R}^3 | i = 1, ..., \ell\}$ but not creating voxels. Unlike RGB images, point clouds have *unordered* property. That is, a set of point clouds doesn't have a specific order. As a result, a network must be invariant to permutations of input points. From the perspective of this property, Qi et al. design the Pointnet series netowk[44, 45] to handle the problem.

3.2.1 Pointnet

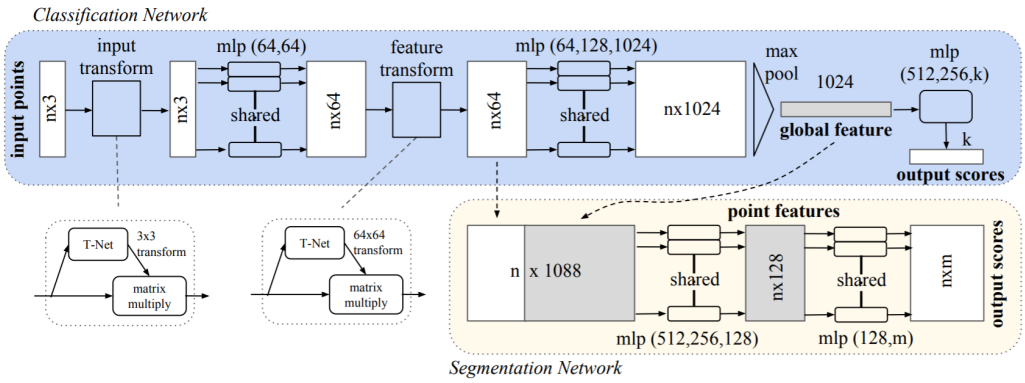


Figure 3.2: Architecture of Pointnet.

Pointnet is the first work in the Pointnet series. In this work, they try to perform classification and segmentation on a set of point clouds. The whole design is in Fig. 3.2. Unlike performing 2D convolution on an image, they use 1D convolutions (MLP layers in the image) to extract features individually, avoiding considering non-relative points. At the end of feature extracting, they use a single symmetric function, max pooling, to decide the final global feature. Therefore, no matter what the order of input is, the network can extract the most representative feature.

doi:10.6342/NTU202100293

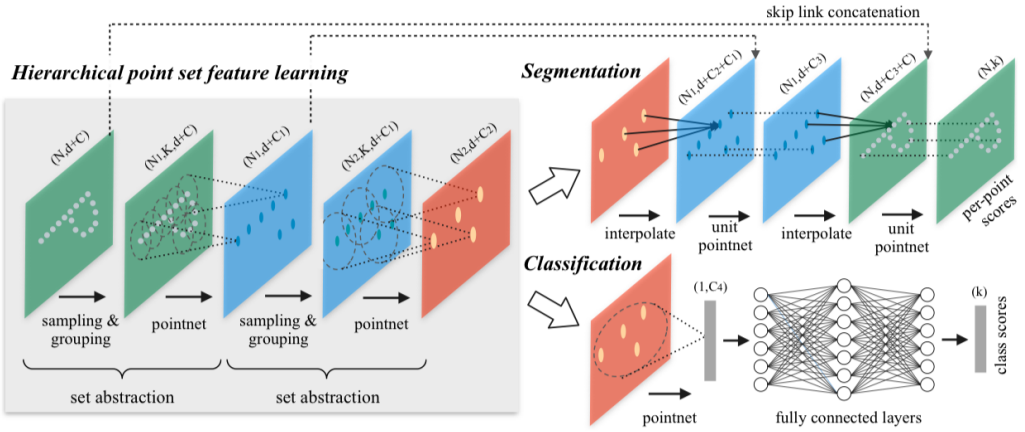


Figure 3.3: Architecture of Pointnet++.

3.2.2 Pointnet++

In Pointnet++, the authors try to solve a problem: how to explore the local features. To this end, they propose a hierarchical module named set abstraction (SA). As shown in Fig. 3.3, the main idea of SA is sampling and grouping. They first uniformly sample several points as centers and then group input points into several clusters. For each cluster, it will go through a simple Pointnet to extract local a feature. Therefore, the local structures are encoded hierarchically by several stages of SA modules.

3.3 3D Joint Detection

Given the a depth image of a target and the corresponding 2D joints, we project them back to 3D camera space, forming point clouds and initial 3D joints. We then use them as input data to estimate the corresponding 3D joint location in camera coordinate. As mentioned in the previous section, we exploit the design of Pointnet and Pointnet++. In this stage, we will describe two different models we design. The first one is the final version of our proposal that utilizes LJN_{jd}. The second one is the basic model, and it's also our baseline model, which is used to prove the

doi:10.6342/NTU202100293

effectiveness of LJN$_{jd}$.

3.3.1 Proposed Model

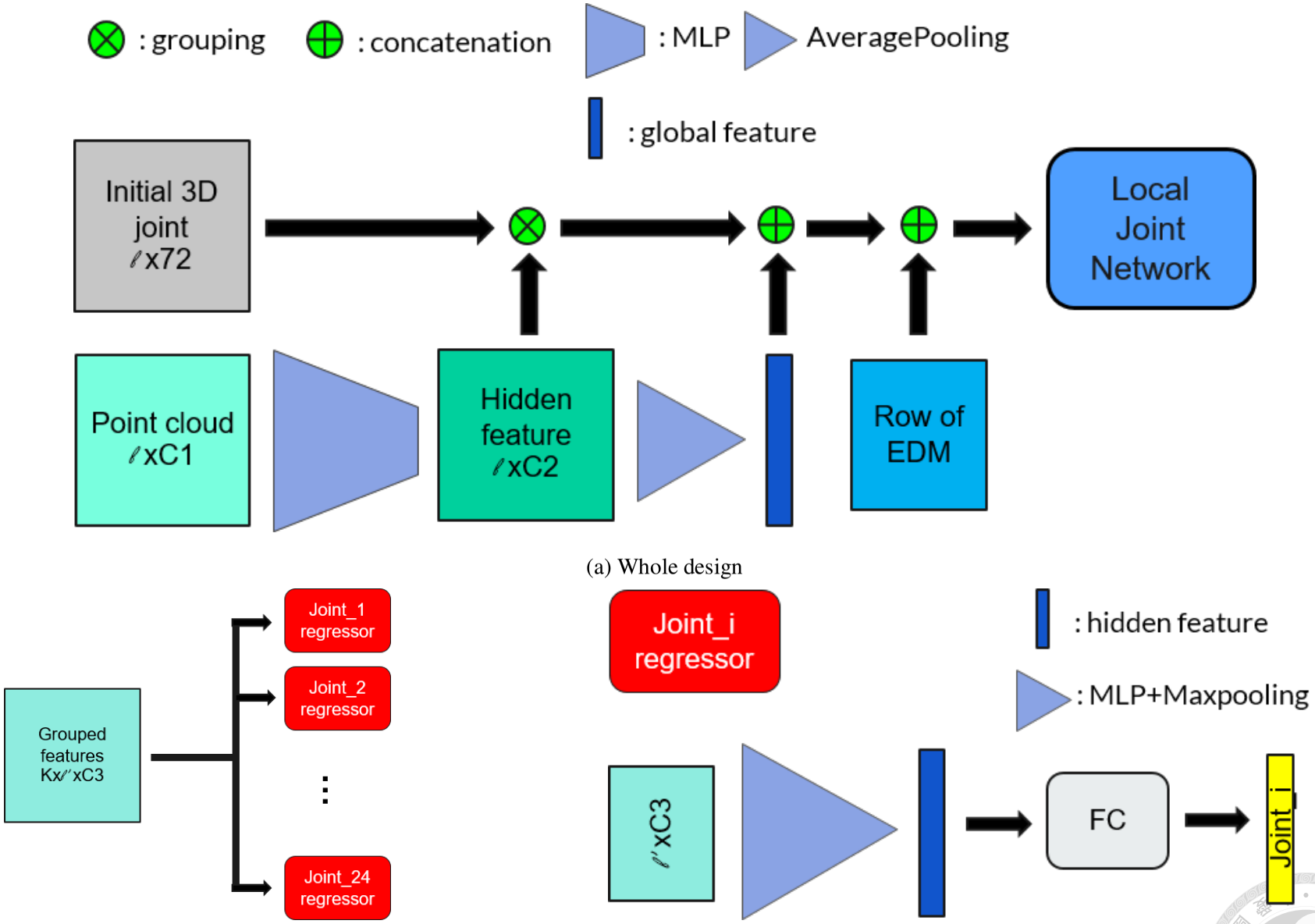
In this model (Fig. 3.4), we extract features with Pointnet first and then use LJN$_{jd}$ to predict joint location. The main idea of LJN$_{jd}$ is that it focuses on the local part of every joint. LJN$_{jd}$ consist of K sub-networks. Each of them takes a group of points as input, which is constructed by gathering ℓ' points around the initial joint location in a given ball query with a specific radius. We name the operation "grouping." For example, i-th input of i-th sub-network is a group of points centered around the i-th joint. Therefore, every sub-networks in LJN$_{jd}$ can pay attention to the joint it belongs to.

Furthermore, we adapt two schemes to strengthen the global relationship between each point and between each joint. First, we perform the grouping operation on the hidden features before the pooling layer and concatenate the global feature after passing the pooling layer. This arrangement is also used in Pointnet. Second, we add the row of the euclidean distance matrix of initial 3D joints to each group, e.g., the group of i-th joint keeps the i-th row (or column) of EDM. Because a joint is not entirely independent of any other joints, we let each sub-network in LJN$_{jd}$ infer other joints' information by the distance matrix.

3.3.2 Basic Model

Except the proposed model, we also design a basic model to prove that our LJN$_{jd}$ is effective. Fig. 3.5 shows the architecture of basic model. In this design, we simply concatenate the initial joint location with input points, forming a input data with $\ell \times (C1 + 72)$ features. We use a Pointnet-liked design to extract features from the input and directly regress all joint locations.

doi:10.6342/NTU202100293



(a) Whole design

(b) Local Joint Network for detection

(c) Sub-network

Figure 3.4: Architecture of proposed model of 3D joint detection. (a) This shows the whole architecture. Hidden features are grouped according to the initial joints. i-th group is concatenated with a common global feature and i-th row of EDM. (b) LJN_{jd} is made up of K sub-networks named joint regressor. (c) This reveals the detailed design of a joint regressor. The architecture is the same as the classification part of Pointnet.

doi:10.6342/NTU202100293

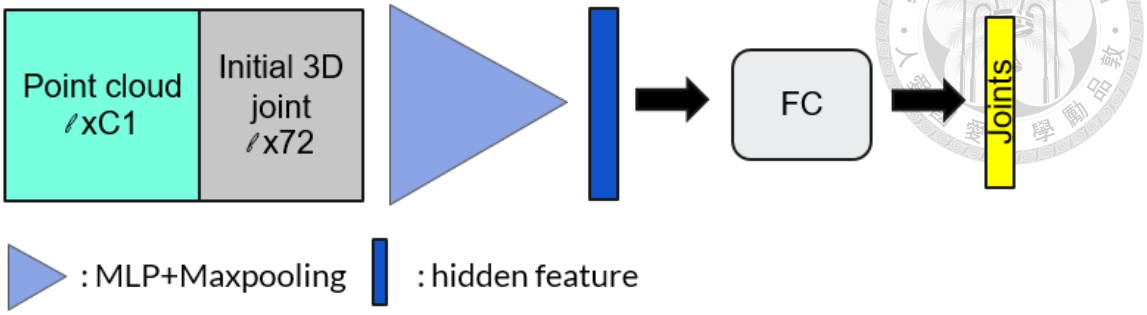


Figure 3.5: Architecture of basic model of 3D joint detection. In this model, we don't use the grouping operation but straightly regress all joint location. The feature extractor (blue triangle) is a similar design as Pointnet. The output dimension is 72 (24×3).

3.4 Human Pose and Shape Estimation

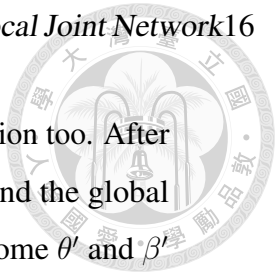
Given a depth map of a target and it's 3D joint location, we aim to predict the corresponding human mesh in SMPL format. We also propose two models. The first one is the basic model which is direct regression. The second one is the proposed model that use LJN$_{pse}$.

3.4.1 Proposed Model

As illustrated in Fig. 3.6, we predict initial parameters first with Pointnet++ and then refine them with LJN$_{pse}$. Our LJN$_{pse}$ is similar to HKMR[15]. HKMR uses chain-based regressors, which correct the pose parameters for a parent trunk first and then send the parameters to the child trunk, to predict the SMPL parameters sequentially. Different from them, our LJN$_{pse}$ uses joint-based regressors. That is, we refine a single joint (parent joint) first and send it to its child joints that it can reach. The parent-child relationship between joints can be found in Fig.3.8.

Before giving an example, we clarify the symbols used here first. We denote θ for the whole pose parameter, θ_i for the pose parameter of i-th joint, β for the shape parameter. A symbol with an apostrophe means it is from the previous stage. As shown in Fig. 3.7a, when we are updating θ'_i, all refined θ_j of the parent joints

doi:10.6342/NTU202100293



that can be traced, β', and the global feature are taken into consideration too. After θ is processed, the beta regressor then regresses β based on β', θ, and the global feature. After getting θ and β, one iteration is over and θ and β become θ' and β' for next iteration. LJN_{pse} repeats the above pipeline N times to output the final prediction.

3.4.2 Basic Model

We also have a basic model to show that our LJN_{pse} is practical. The whole architecture are in Fig. 3.9. In this desing, we also use Pointnet++ to extract the same features as the proposed model. However, we use those features to directly predict all the residual values of SMPL parameter. We will show that this direct regression degenerate the performance in Section 3.7.

3.5 Data Preparation

In our settings, we need depth maps and corresponding SMPL ground truths. However, parametric model for an individual doesn't exist ground truth and most of datasets don't come with depth maps. Therefore, we conduct our experiments on synthetic data generated by ourselves. Like [6], we use the SMPL parameters provided by SURREAL[46] and render a set of depth maps.

3.5.1 Data Sampling

Learning from Synthetic Humans (SURREAL) is a large-scale synthetic human dataset. They fit CMU MoCap database [47] into pose parameters of SMPL and sample human mesh from CAESAR dataset [48] to acquire shape parameters. They generate more than 6 million data (see Table 3.1 for more details). Each is provided with an RGB image, 2D and 3D joint label with SMPL joint definition, SMPL parameter and mesh, segmentation map, depth map, and optical flow. They also offer 3 different overlap ratios for choice.

doi:10.6342/NTU202100293

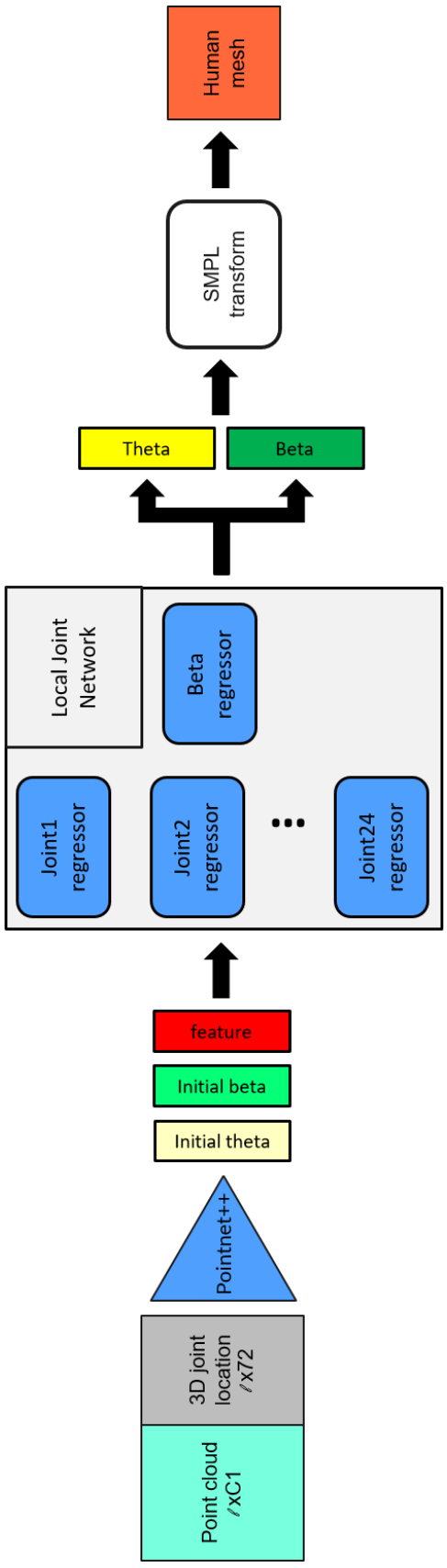
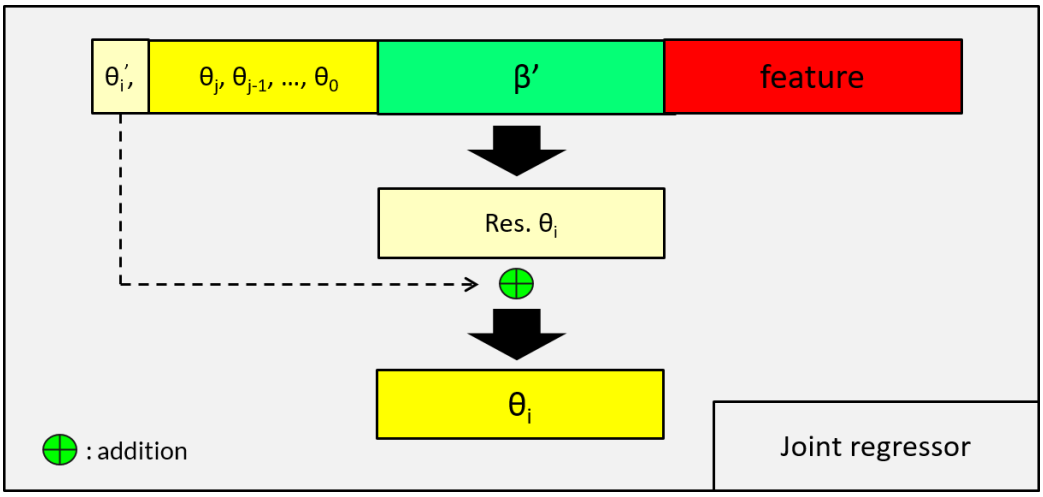
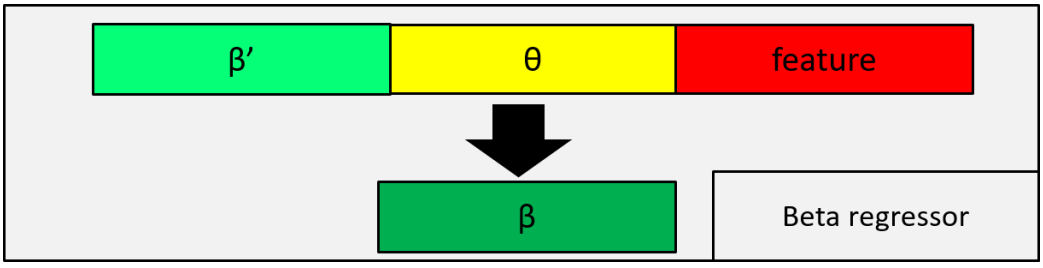


Figure 3.6: Overall architecture for human pose and shape estimation. We use Pointnet++ to extract a global feature (red one) and an initial SMPL parameter.The parameter and the global feature then are sent to a hierarchical regressor to predict final parameter.

doi:10.6342/NTU202100293



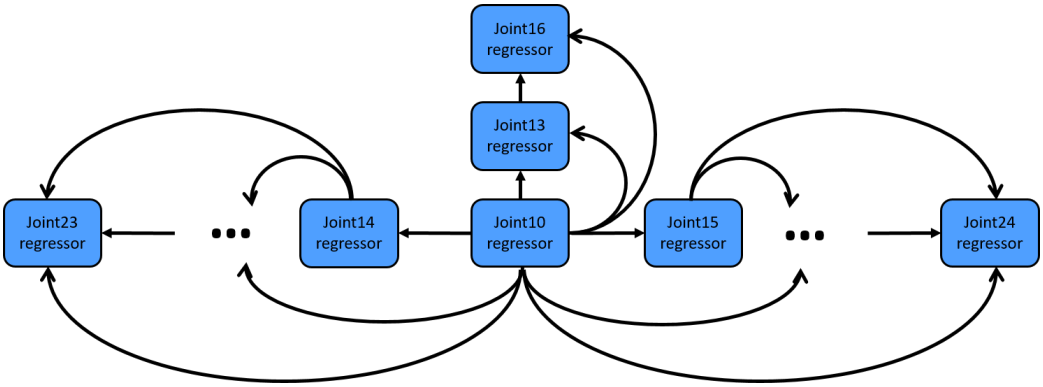
(a) Joint regressor



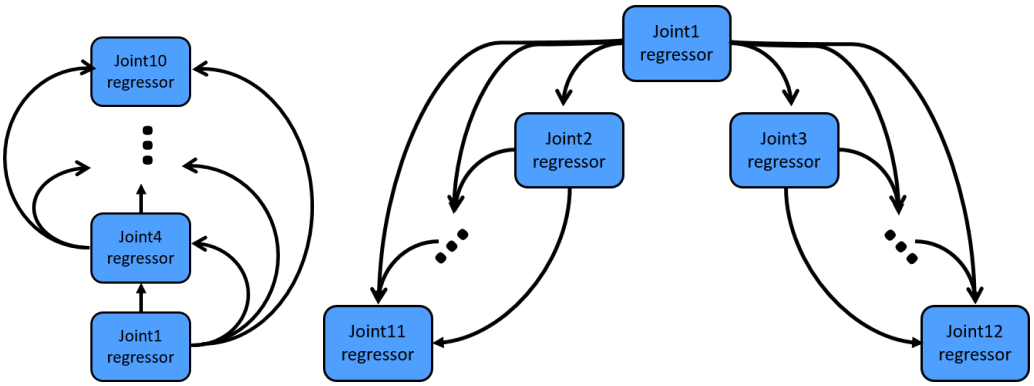
(b) Beta regressor

Figure 3.7: Joint regressor and beta regressor. The θ_i in (a) represent the pose parameter of i-th joint. In an iteration, joint regressor will first find the residual of each θ_i, and then refine β. The sub-labels with j in (a) are not the real labels. They means the parent joins that can be traced from the i-th joint.

doi:10.6342/NTU202100293



(a) Upper body



(b) Middle body (c) Lower body

Figure 3.8: The connection between each joint regressor. We separate the whole body into 3 parts to visualize clearly. Every joint except the endpoints has at least one child joint. The theta of i-th joint will be transmitted to the joints it can reach.

doi:10.6342/NTU202100293

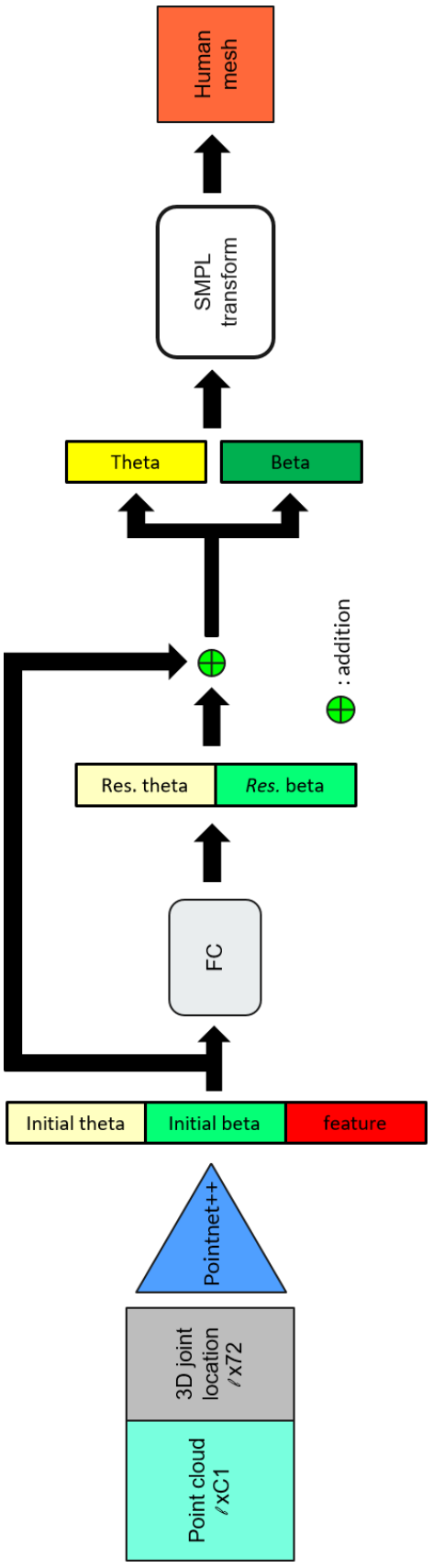


Figure 3.9: The basic model of human pose and estimation.

doi:10.6342/NTU202100293

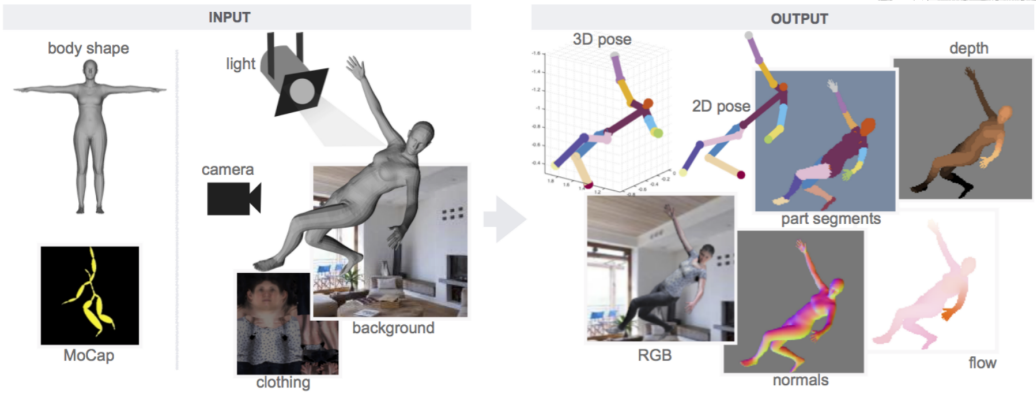


Figure 3.10: Data generation pipeline in SURREAL.

	#subjects	#sequences	#clips	#frames	#our samples
Train	115	1,964	55,001	5,342,090	60,000
Validation	-	-	-	-	5,000
Test	30	703	12,528	1,194,662	15,000

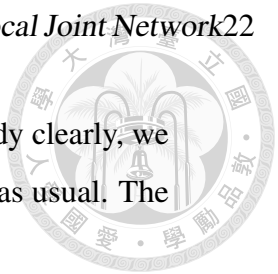
Table 3.1: Numbers of data. The validation set is a sub-set of testing set. The authors don't mention it in the original paper but it is available on their website.

We choose one overlap ration (run0 named by the authors) and sample at least 3 data from the clip more than five frames. We finally generate 60 thousand training samples, 5 thousand samples for the validation set, and 15 thousand testing samples. Specially, we only use the SMPL parameter. We transform those parameters in meshes and render new sets of depth maps. Next, we will discuss how we generate our data.

3.5.2 Data Rendering

We use Kinect v2 camera settings to render our data. Kinect v2 has one RGB camera with resolution $(1920, 1080)$ and one time-of-flight (ToF) camera with resolution $(424, 512)$. We only use the data of the ToF camera. However, we set

doi:10.6342/NTU202100293



the resolution to $(512, 424)$. That is, to capture the whole human body clearly, we assume the Kinect camera is placed vertically but not horizontally as usual. The intrinsic parameters $\mathcal{K}$ we use is as followed:

$$\mathcal{K} = \begin{bmatrix} 366.458 & 0 & 206.470 \\ 0 & 366.736 & 253.026 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Another problem is that the global rotation (rotation of the root joint) of the parameters get from SURREAL is random. In other words, we need to decide the rotation matrix of the extrinsic parameter additionally. After observing visualized data, we fond that most of the data can be transformed to "*front view*" (see Fig. 3.11 for the definition of *front view*) by a common rotation. Therefore, the rendering procedure can be formulated into the follow equation:

$$D = \Pi(R \cdot M + T) \tag{3.4}$$

$$R(M) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{3.5}$$

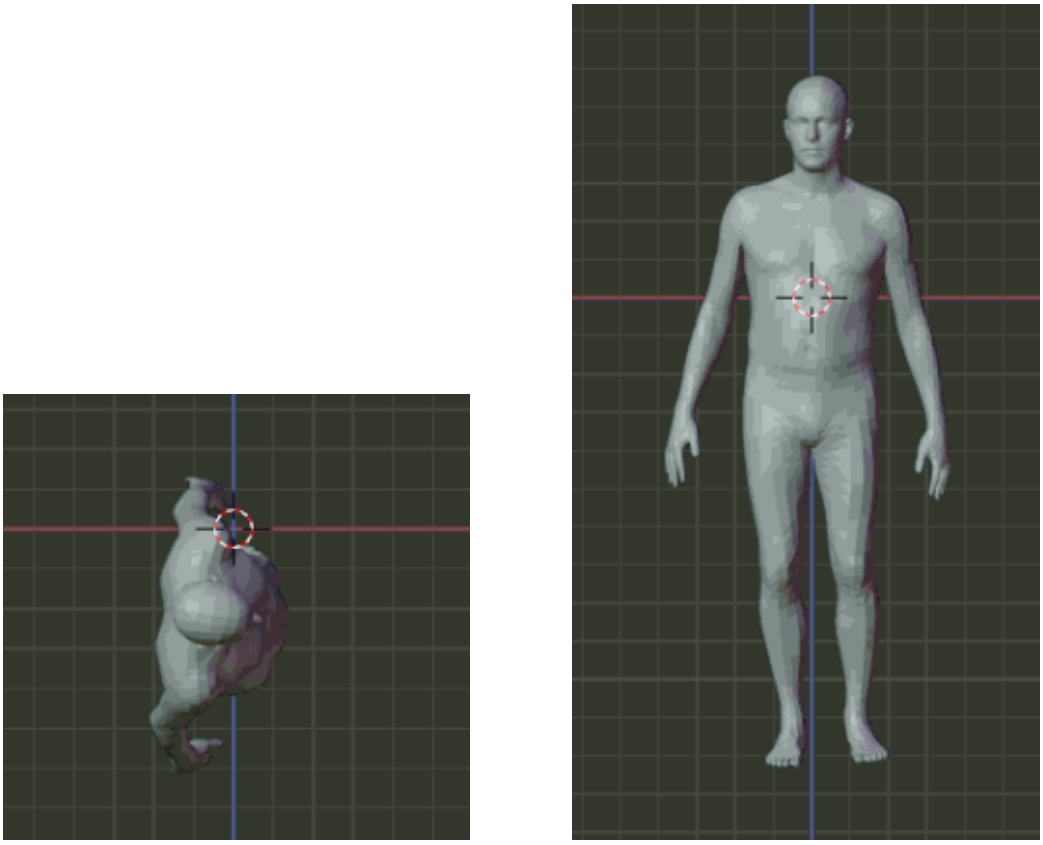
D is the depth map of a mesh M. Π is the projecting function in OpenGL with intrinsic parameter $\mathcal{K}$. R is the fixed rotation matrix and T is random translation in the extrinsic matrix.

3.6 Implement Details

3.6.1 Common Settings

All depth maps are segmented by a mask and added Gaussian noises with zero mean and 0.001 standard derivation before transformed to point clouds. We also use normal vectors calculated by Open3D[49] as additional input features. We use RAdam [50] optimizer with default settings. The learning rate decays 0.1 every 5

doi:10.6342/NTU202100293



(a) Original view (b) front view

Figure 3.11: Different view of a mesh. Most of the meshes get from SURREAL are looked like (a), or "*top view*." We define the "*front view*" as in (b). The position of head should be the highest except that the action of the target is something like crawling. We find that by a fixed rotation, most of the meshes can be rotated to "*front view*."

doi:10.6342/NTU202100293

epochs. Because we use small batch size in training stage, we remove all batch normalization layer in the original Pointnet and Pointnet++. Additionally, we use PReLU[51] as activation layers but not ReLU.

3.6.2 3D Joint Detection

We simulate disordered initial joints by adding Gaussian noise. We totally simulate 4 different levels by changing standard deviation σ. The unit simulated noise is pixel. The number of sampled points ℓ' of a sub-network is 128. Searching radius $\mathtt{r}$ of each sub-network is different. The loss function is:

$$loss = \sum_{i=1}^{K} \left\| \tilde{J}_i^{3D} - J_i^{3D} \right\|_2^2 \tag{3.6}$$

$\tilde{J}_i^{3D}$ is the predicted 3D joint location and J_i^{3D} is the ground truth location.

3.6.3 Human Pose and Shape Estimation

We have two different initial joints settings. The first one is we use the estimated result from the proposed model in Section 3.3.2. The other one is we add Gaussian noise with different standard deviation to ground truth joints. The unit of simulated noise is meter. For loss function, we don't calculate loss on parameters. The direct supervision on parameters is meaningless, leading to over-fitting on training data. Therefore, we transform the parameters to human meshes and then calculate the loss:

$$loss = \frac{1}{\mathcal{V}} \sum_{t=1}^{\tau} \sum_{i=1}^{\mathcal{V}} \|\tilde{v}_{t,i} - v_i\|_2^2 \tag{3.7}$$

$\tilde{v}_i$ is a vertex of the mesh transformed from the predicted theta and beta and v_i is a vertex of ground truth mesh.

doi:10.6342/NTU202100293

3.7 Result

3.7.1 3D Joint Detection

Evaluation metrics

We use two metrics to evaluate our models: mean per-joint position error (MPJPE) and percentage of correct key-points (PCK). MPJPE is as in Equation 3.8, which is the mean L2 distance over all joints and samples. The unit is millimeter (mm). PCK counts the percentage of correct key-points, whose error are less than a certain threshold. We choose the threshold as half of head bone length (ground truth distance between 13-th joint and 16-th joint). We denote it **PCKH@0.5** in our tables. The unit is percentage ($\%$).

$$MPJPE = \frac{1}{NK}\sum_{t=1}^{N}\sum_{i=2}^{K}\left\|\tilde{J}_{t,i}^{3D} - J_{t,i}^{3D}\right\|_2 \tag{3.8}$$

Comparison between each model

We first evaluate the performance of the two designs. As in Table 3.2, we can find that the basic model is good enough. Nevertheless, the proposed model boosts the accuracy by almost 8 mm. This result indicates that our grouping strategy and LJN are effective and powerful. Each sub-network can focus on the local region. Moreover, the sub-network can easily obtain global information from the global feature. In contrast, the basic model needs to put lots of effort into disentangling each part of input points, leading to a higher error. We also evaluate the function of adding rows of EDM. The same table shows that the performance is boosted further. This result verifies that a joint's sub-network requires some prior knowledge of other joints to locate the correct joint location better.

doi:10.6342/NTU202100293

	MPJPE	PCKh@0.5
basic	32.1	85.6
proposed w/o EDM	25.0	90.2
proposed	**24.2**	**90.8**
simplebaseline[4]($\sigma = 0$)	59.4	51.9
simplebaseline[4]($\sigma = 10$)	81.3	33.6

Table 3.2: The results of different designs. Our results use $\sigma = 10$.

std of gaussian	max dist.	mean dist.	MPJPE	PCKh@0.5
$\sigma = 0$	0	0	14.3	97.2
$\sigma = 5$	26.6	6.3	18.6	94.7
$\sigma = 10$	53.2	12.5	24.2	90.8
$\sigma = 15$	83.7	19.0	31.4	85.6

Table 3.3: The results of proposed model given different level of disordered initial joints. The unit is millimeter. $\sigma = 0$ means we use ground truth.

Input noise of 2D joints

Table 3.3 is the ablative study of our model under different levels of noise of 2D joints. The 4 standard deviation are 0, 5, 10, and 15. $\sigma = 0$ is equals to ground truth joint. We also provide the maximum value and the mean L2 distance (”max dist.” and ”mean dist.” in the table) of the noisy 2D joints in pixels for better understanding the variation. If the input noise is relativly small (σ is less or equal to 10), our model is stable. The changing of MPJPE and PCK is about 10 mm and 7 percent. In contrast, the performance of our proposed model will degrade much only when noise of input 2D joints is very large ($\sigma = 15$). To model a proper results of 2D detector, we choose $\sigma = 10$ in other tables.

doi:10.6342/NTU202100293

sample points	MPJPE	PCKh@0.5
$\ell = 2048$	25.8	90.1
$\ell = 4096$	24.2	90.8

Table 3.4: The results of different numbers of sampled points ℓ.

Sparsity of input points

We analyze the sparsity by changing the number of sampled points ℓ. As in Table 3.4, we use two different levels. To extracting rich features, we set $\ell = 4096$ as standard. The other one is half of the standard. We can see that when the input resolution is lower, our model don't degrade much. This shows that our model is robust to different input resolution.

Comparison with simple baseline

We further compare our proposed model with [4] ("simple baseline" in Table 3.2), which also starts from 2D joints. We train their network with our dataset and evaluate two different input noise. From the table, we can see that our model out-perform them under the same circumstances. We argue that [4] is confused by the ambiguity of 3D-2D projection. By eliminating the uncertainty with depth prior, our model can focus on localization in 3D space. Furthermore, their model is unable to handle large-scale dislocated 2D joints.

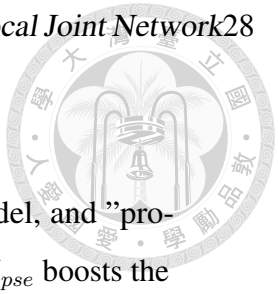
3.7.2 Human Pose and Shape Estimation

Evaluation metric

We use L2 distance to calculate averaged per-vertex error (APVE) without further alignment as in Equation 3.9. We also report MPJPE of meshes' joints.

$$APVE = \frac{1}{N\mathcal{V}} \sum_{t=1}^{N} \sum_{i=1}^{\mathcal{V}} \|\tilde{v}_{t,i} - v_{t,i}\|_2 \tag{3.9}$$

doi:10.6342/NTU202100293



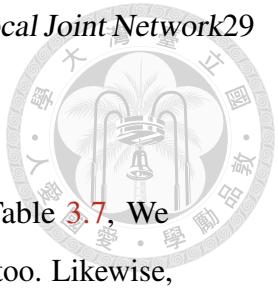
Comparison between models

Table 3.5 shows the results of our models. "basic" is the basic model, and "proposed" is the proposed model using LJN_{pse}. We can see that our LJN_{pse} boosts the performance for about 10 mm, proving that our per-joint design is effective. We also evaluate the hierarchical design in LJN_{pse}. *single* in the table represents that we don't transmit a joint to its child joint (like the sub-network design in chapter Fig. 3.4), and its error is higher than the final proposed for about 8 mm. This result indicates that the hierarchical design following the human kinematic tree is useful. The next comparison is the choice of feature extractor. We believe that the model needs detailed features to estimate the parametric model. Therefore, we change Pointnet++ to Pointnet in *pointnet* in the table, and it proves. Pointnet++ can further increase hte performance.

Effectiveness of noised 3D joints

We study how 3D joints affect the accuracy by changing the standard deviation of Gaussian noise. We evaluate 4 different σ as in Table 3.6. $\sigma = 0$ means it uses ground truth 3D joints as input. We also provide the mean and the maximum L2 distance ("mean dist." and "max dist." in the table) of the noisy 3D joints in millimeter for better understanding the variation. The table shows that our proposed model is reasonably well when using ground truth 3D joints. The proposed model degenerates 6 mm when the input noise is large, which indicates that our proposed model is stable for different level of noise. In the all of our experiments, we choose $\sigma = 0.02$ if there is no further notation. The result using the output of our proposed 3D joint detector is labeled "end-to-end" int the table. Even though it's closed to the result of using $\sigma = 0.03$, the difference between ground truth and it is still less than 10 mm. This demonstrates that our proposed model is stalbe to unknown noise.

doi:10.6342/NTU202100293



Sparsity of input points

We also evaluate the sparsity of input points by changing ℓ. In Table 3.7, We evaluate two different input resolution. The standard one it $\ell = 4096$ too. Likewise, our model degenerate less when the resolution is half ($\ell = 2048$). This result also proves that our model is robust to input sparsity.

	APVE	MPJPE
basic	43.1	42.9
proposed-*single*	42.2	40.6
proposed-*pointnet*	35.6	36.8
proposed	**34.5**	**33.0**
HMR[13]†	54.3	-
Wei et al.[52]†	58.6	-
HMR[13]‡	56.8	87.9
SPIN[14]‡	41.1	-
HKMR[15]‡	-	71.0

Table 3.5: The results of different designs. *pointnet* means it use Pointnet but not Pointnet++ as the feature extractor. *single* represents that LJN$_{pse}$ is non-hierarchical design.The result with † means it comes from Wang et al.[6]. The result with ‡ means it is the result on RGB dataset Human3.6m[7] reported in the original paper and use protocol 2 when calculate MPJPE.

doi:10.6342/NTU202100293

std of gaussian	max dist.	mean dist.	APVE	MPJPE
$\sigma = 0$	0	0	31.6	30.5
$\sigma = 0.01$	52.9	12.5	32.9	31.5
$\sigma = 0.02$	105.6	25.1	34.5	33.0
$\sigma = 0.03$	170.0	37.6	37.6	36.0
end-to-end	-	-	38.8	36.8

Table 3.6: The results of proposed model given different level of disordered initial joints. *end-to-end* use the 3D joints from our proposed 3D joint detector with $\sigma = 20$ and $\ell = 4096$ to ensure both joint detection and pose and shape estimation have enough performance.

sample points	APVE	MPJPE
$\ell = 2048$	36.0	34.3
$\ell = 4096$	34.5	33.0

Table 3.7: The results of different numbers of sampled points ℓ.

doi:10.6342/NTU202100293

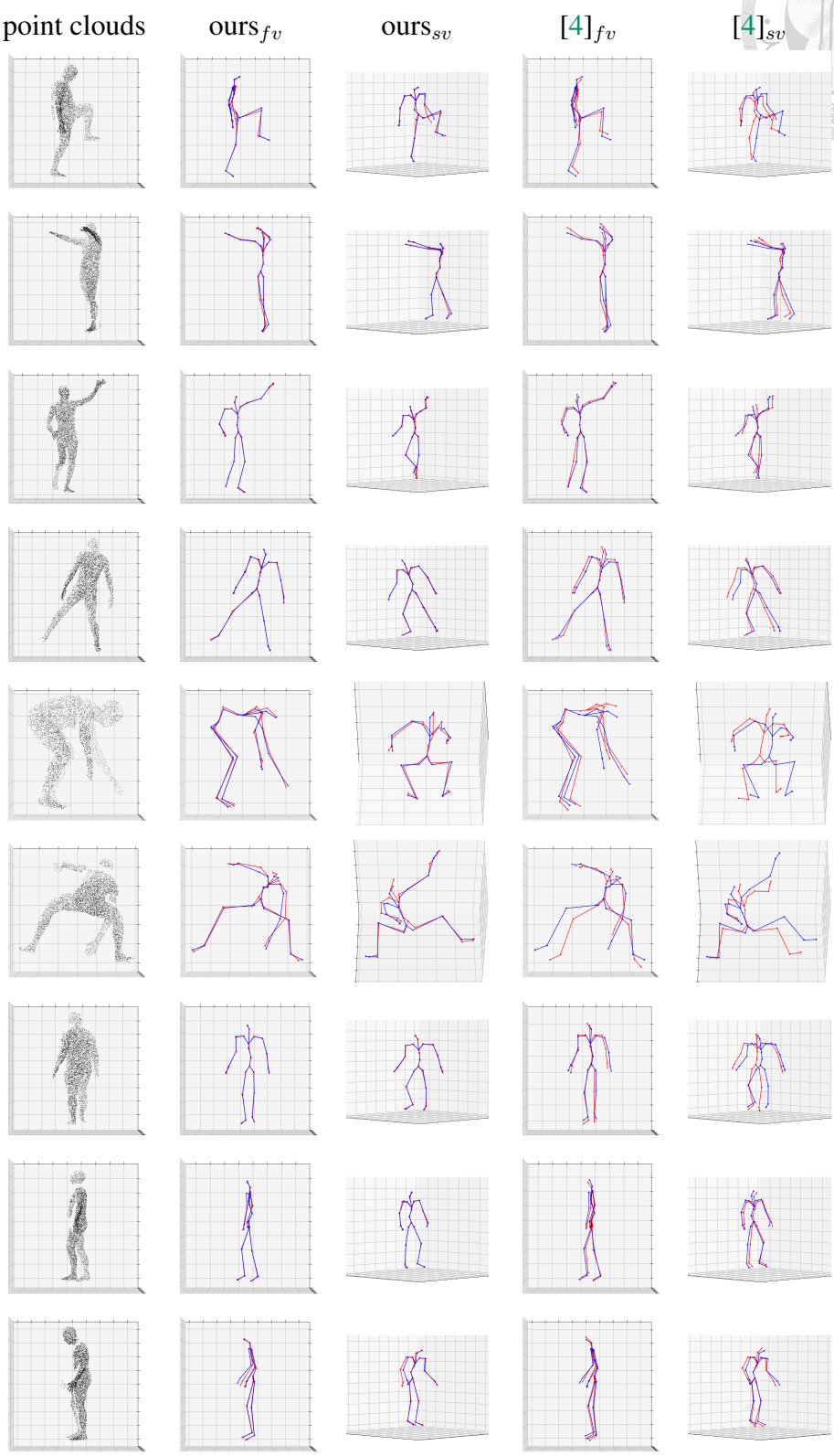


Figure 3.12: Visualization. The number of input points is sampled to 4096. The blue skeletons are the ground truth, and the red ones are the predicted. We draw two views, fv for front view and sv for side view, for better illustration. fv is the same view as input point clouds, and sv is set by rotating the whole graph properly. "ours" are the results from the proposed model with $s = 5$. For [4], we use $s = 0$.

doi:10.6342/NTU202100293

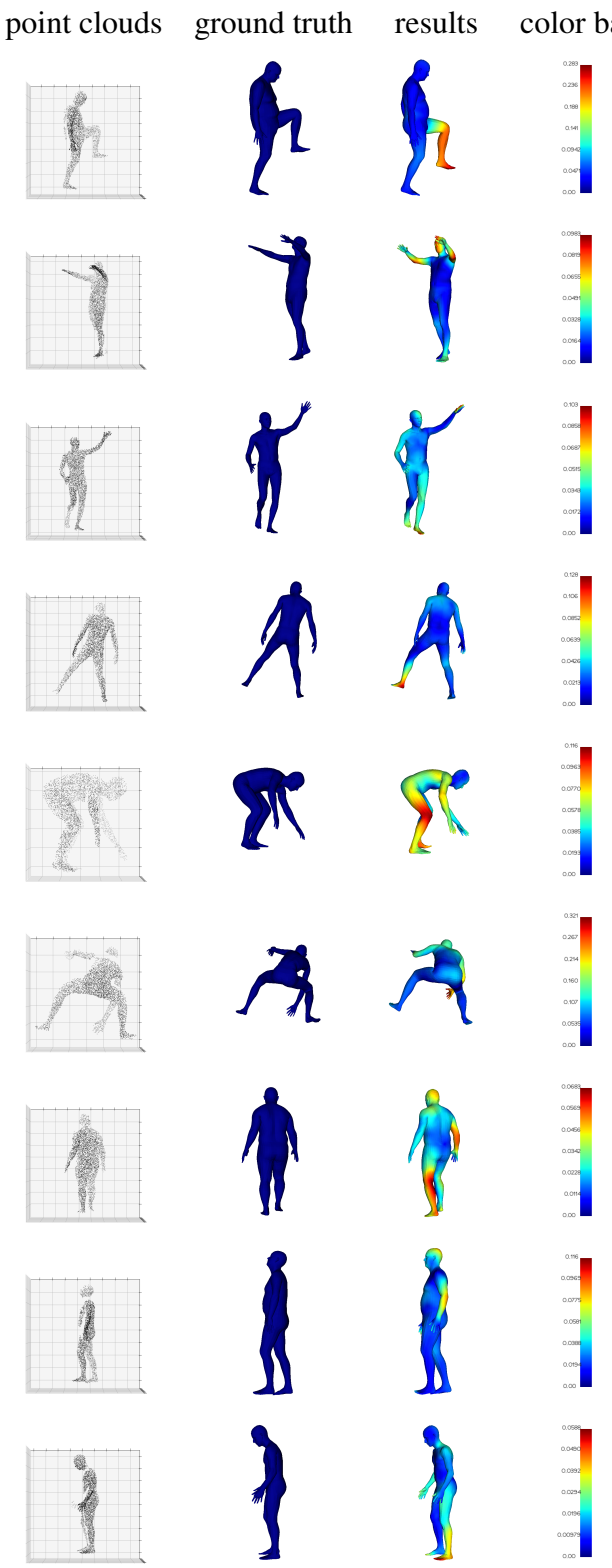


Figure 3.13: Visualization.

doi:10.6342/NTU202100293

Chapter 4

System

4.1 Introduction

As mentioned in Chapter 5, we accomplish an augmented reality system by merging our two proposed models. Except for the synthetic data used in Chapter 3, we also collect some real data and test the system on it. Although we train our models with synthetic data, we find them efficient enough to be directly used in a real scene. The main challenges are: 1) How to get confident 2D joints. 2) We need to align the output meshes with images. To this end, we train a particular CPN to get 2D joints in SMPL format and design a procedure to manage the alignment problem. The full pipeline is shown in Fig. 4.1. In the following sections, we will first discuss the settings of our system and the main architecture. The visualization of the output of the system is at the end.

4.2 Experiment Settings

4.2.1 Scene

We assume the input source is Kinect v2 camera. The distance between the camera and the target is about $1.5 \sim 2$ meters to ensure that the whole body can be

doi:10.6342/NTU202100293

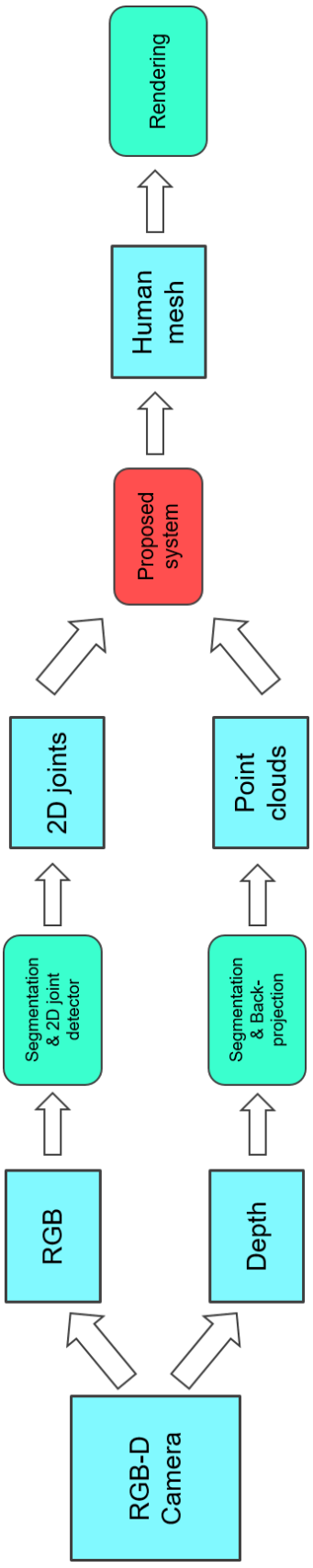


Figure 4.1: The pipeline of our system. Both the RGB image and the depth image share the same segmentation mask.

doi:10.6342/NTU202100293

captured.

4.2.2 Data Preparation

We capture several real scenes with a Kinect v2. The resolution $(height, width)$ is $(512, 424)$. The RGB images are also registered to the same resolution. We put a simple green curtain behind the target to do background removal. The whole segmentation procedure is accomplished by OpenCV[53]. We also use another depth threshold to exclude outlier pixels to make the mask more perfect. Some examples can be find in Fig. 4.3. We also have some results from synthetic data. The settings are the same as the testing set in Section 3.5.

4.3 Proposed system

4.3.1 2D Joint Detection

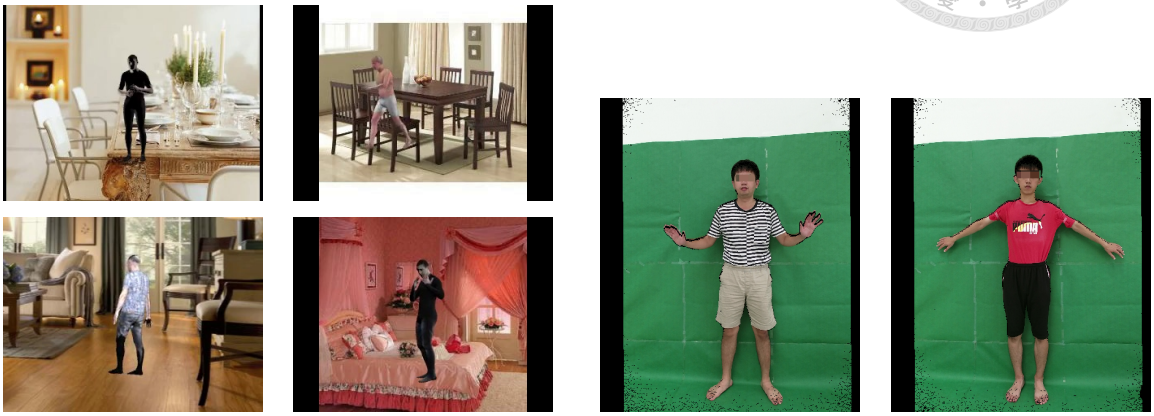
As stated, we use CPN as our 2D joint detector but not train a new one. No matter the accuracy and the speed are efficient enough in CPN. We train a particular version that can predict SMPL joints, which is unreachable for most pre-trained work. The training details are the same as the original paper, except that we use the RAdam optimizer. The training data is the RGB images of SURREAL.

However, the testing result on real data will fail if we use the weights directly trained on SURREAL. We discover that this phenomenon results from the distraction of the background. As shown in Fig. 4.2, an image in SURREAL has an intricate background image irrelevant to the target. In other words, the network put lots of effort into separating the target and the background. Nevertheless, the whole structure of real data is quite different from synthetic data. Accordingly, the network fails on real data.

To address this problem, we use background-removed images in the training stage. We segment images with the masks given by SURREAL and set the back-

doi:10.6342/NTU202100293

ground black. The network then can focus on the target. Fig. 4.3 demonstrates that the proposed solution is good enough for real data.



(a) SURREAL (b) Real images

Figure 4.2: Synthetic data in SURREAL and our real data.

4.3.2 3D Human Mesh Estimation

To get human meshes, we combine the two proposed models in Sec. 3.3.2 and 3.4. That is, we use the 2D joints from the CPN in the previous section as initial joints and predict the 3D joint coordinate in camera space with the first model. After that, the 3D joints are sent to the second model to get the estimated human mesh.

4.3.3 Post Processing

In Section 3.5.2, we mentioned that we use random translation in rendering stage. However, to accomplish this AR system, we need to know the translation to render meshes on the right position. To address this issue, we align the predicted root and the root of the rotated estimated mesh. We assume the predicted root is accurate enough that the translation between the two roots can approximate the real translation, the T in Equation 3.5. Therefore, the rendering procedure is:

$$I = \Pi(R \cdot \tilde{M} + T') \tag{4.1}$$

doi:10.6342/NTU202100293

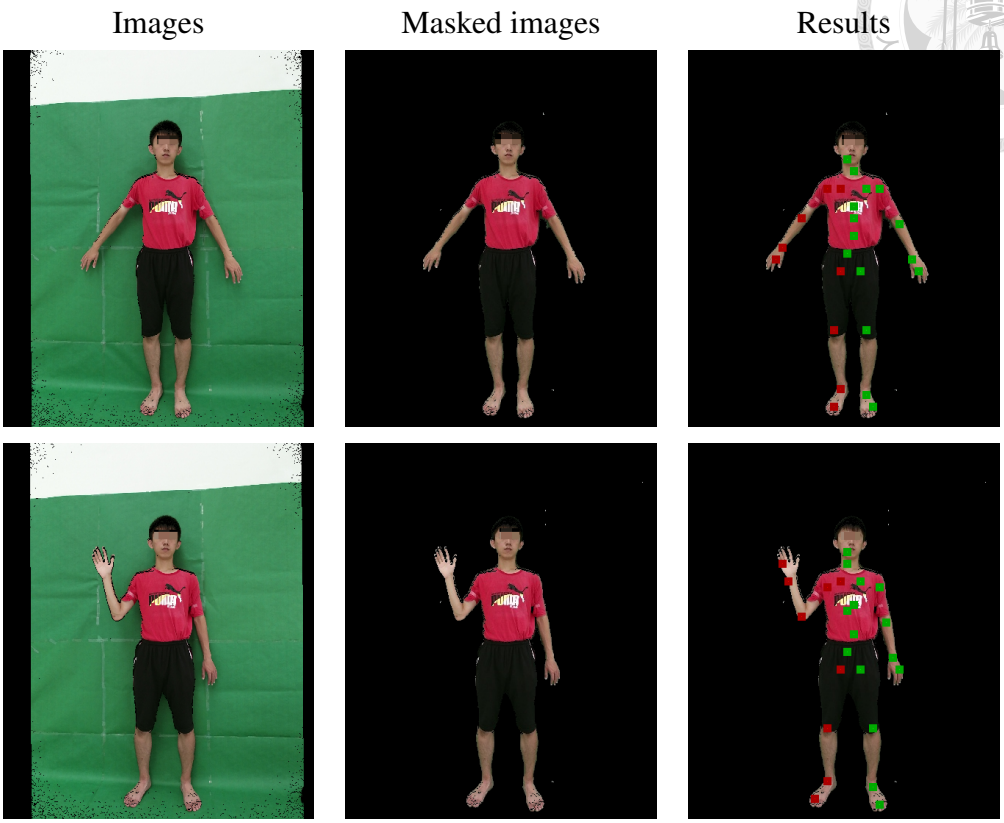


Figure 4.3: Visualization of our CPN results. The lighter dots inside the squares on the **Results** column are predicted joint locations. Color red represents right part of a human body, and color green is the opposite. The direct inference results are reasonably well on real data.

$$T' = \begin{bmatrix} \tilde{J}_{0,x} - \tilde{J}'_{0,x} \\ \tilde{J}_{0,y} - \tilde{J}'_{0,y} \\ \tilde{J}_{0,z} - \tilde{J}'_{0,z} \end{bmatrix} \tag{4.2}$$

I is the rendering result. Π and R are the same as in Equation 3.4. $\tilde{M}$ is the predicted mesh. $\tilde{J}_0$ is the predicted root and $\tilde{J}'_0$ is the root of rotated $\tilde{M}$.

doi:10.6342/NTU202100293

4.4 Experiment Result

4.4.1 Synthetic Data

The synthetic sequences are rendered in gray-scaled, and we only use ground truth mask. The 2D joints are also taken from CPN. In Fig. 4.4, we can see that because our model is trained with synthetic data, the result is ideal. Most frames are matched. However, we still have some failed case. Those failed cases are caused by wrong 2D joints. As in Fig. 4.5, we can discover that the left and the right joints are flipped. Also, some occluded joints are predicted in the opposite part. Therefore, the predicted meshes are wrong because the network consider the target is reversed left-right.

4.4.2 Real Data

We directly test on real data without further fine-tuning. In Fig. 4.6 and Fig. 4.7, we demonstrate the results of two sequences of two different targets. Although the predicted meshes are not matched perfectly, the results is still reasonably well. The motion of predicted meshes are the same input images. We also visualize the 2D joints. We can discover that the results are stable for most frames. Even some 2D joints are particularly noised, the results don't deviate much.

We argue that the error mostly come from the domain gap between real data and synthetic data. In training stage, we use texture-less SMPL meshes to render our data. Furthermore, we use the ground truth masks. Nevertheless, this two conditions are impossible for real data. As a result, our model is confused by the noises, texture and outlier points, and output incorrect parameters.

doi:10.6342/NTU202100293

Figure 4.4: Some frames of results of synthetic sequence.

doi:10.6342/NTU202100293

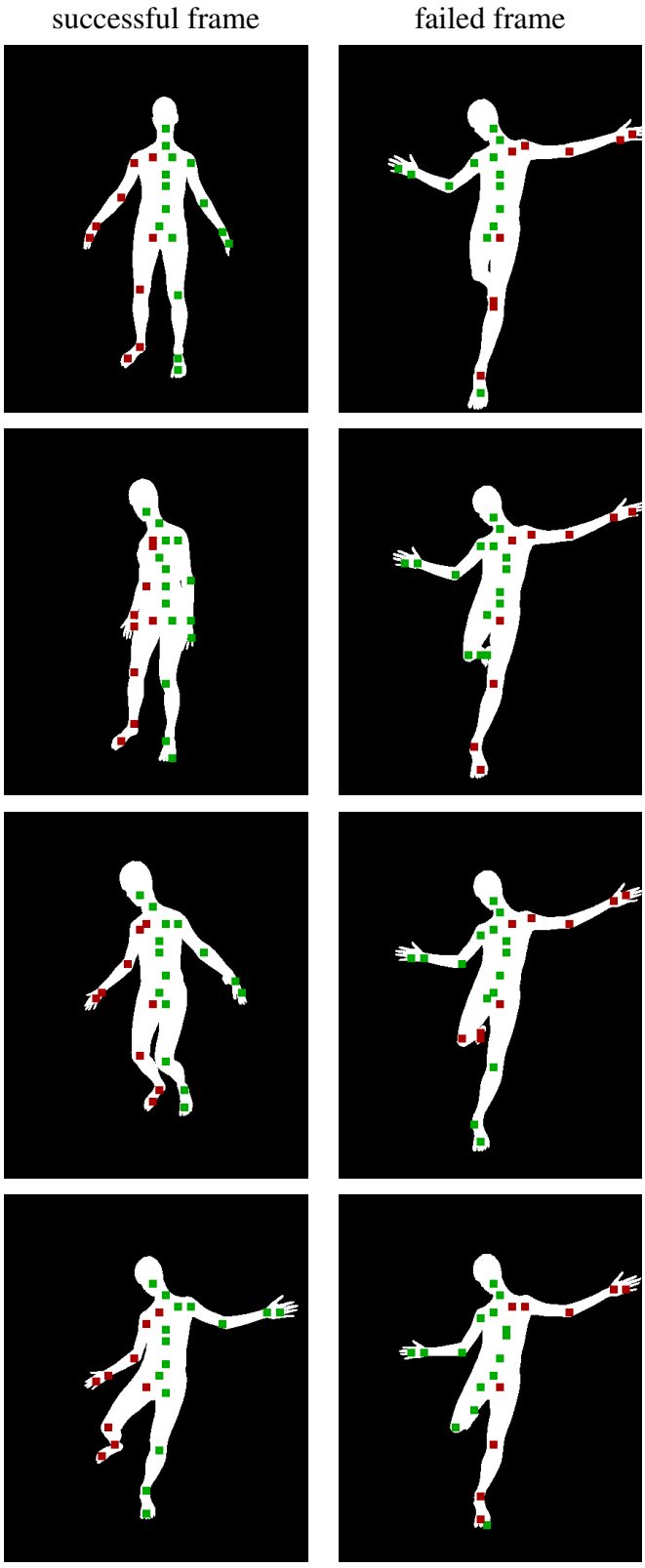


Figure 4.5: Visualization of 2D joints. Color red indicates right part of a human body. Color green is the opposite. We can find that the failures have already failed in 2D detection.

doi:10.6342/NTU202100293

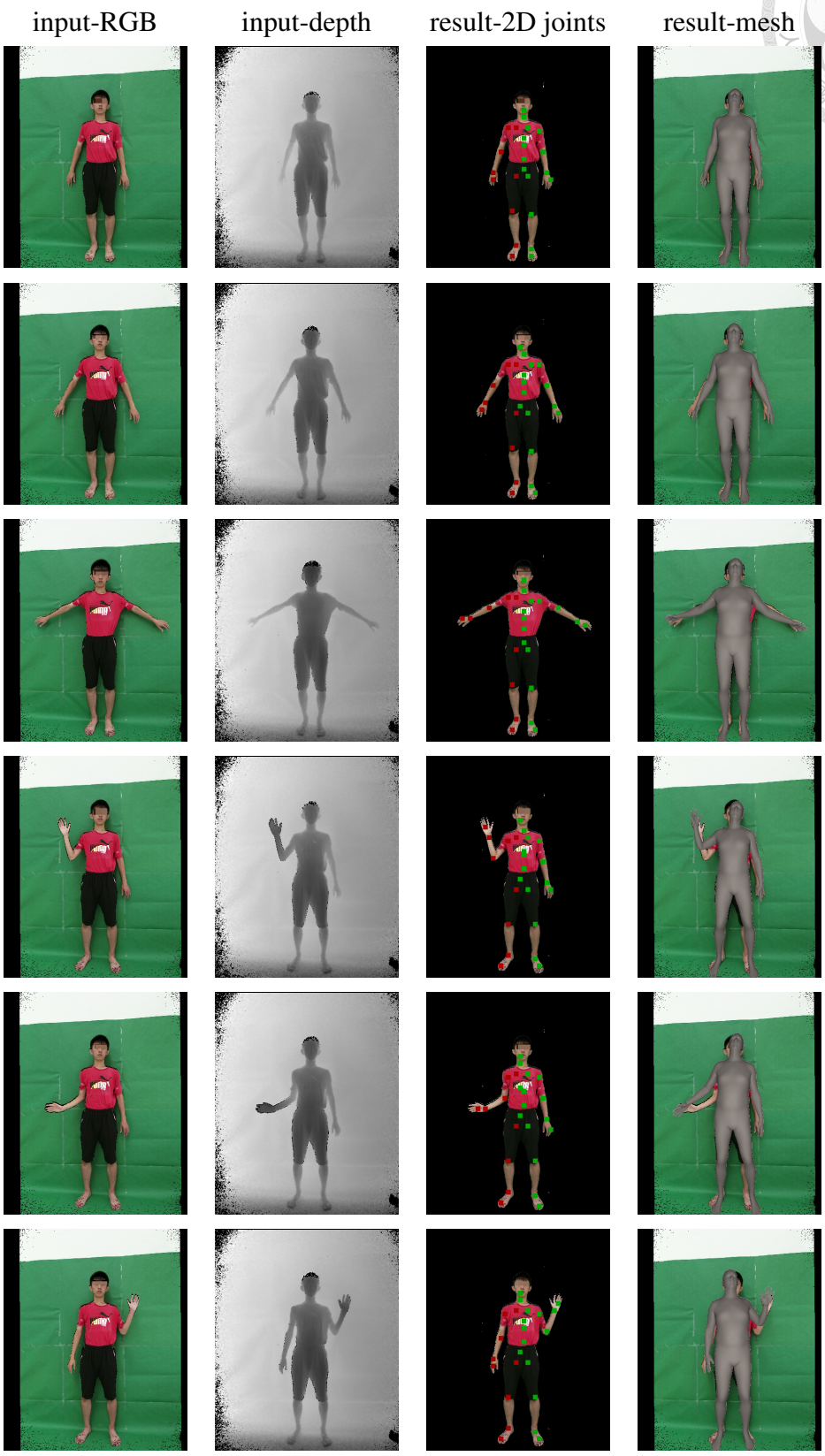
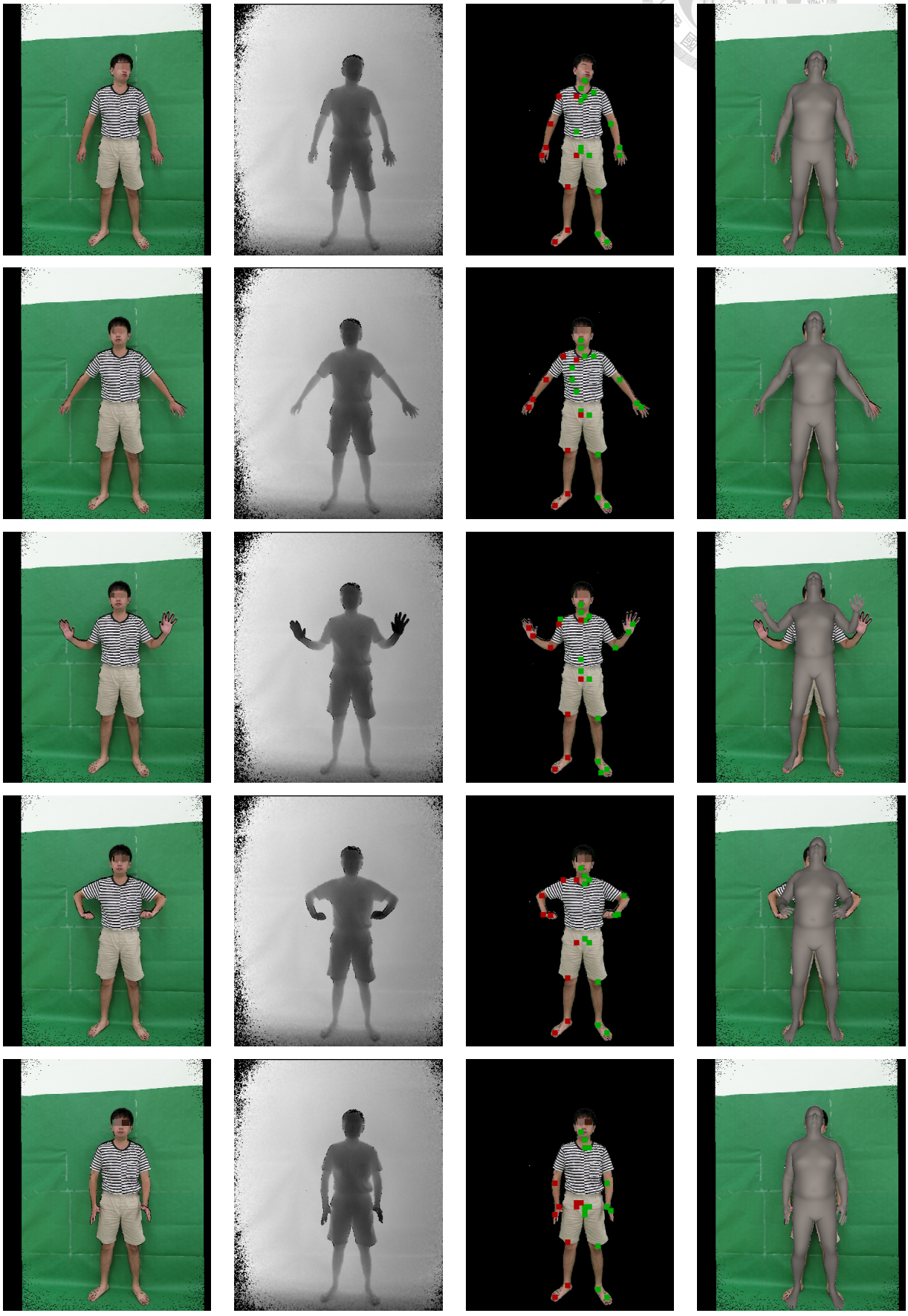


Figure 4.6: Visualization of real data of the first target.

doi:10.6342/NTU202100293



doi:10.6342/NTU202100293

Figure 4.7: Visualization of real data of the second target.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we study how to estimate 3D human joints and human parametric models from point clouds. We propose two novel strategies called Local Joint Network for joint detection (LJN$_{jd}$) and for pose and shape estimation (LJN$_{pse}$). The two networks are aim to process the information of different joints separately without the interference from others. We demonstrate the effectiveness of our methods on synthetic data by considering different settings. Furthermore, we design a flow of an AR system which can output augmented images with human mesh on it. The outputs of the system also prove that our methods works adequately on real data.

5.2 Future Work

Our two models are currently experimental. There still have many improvements. First, we don't use other priors and regularization to regularize valid detection and pose and shape. Adding proper priors or regularization can make the result more realistic. Second, we don't consider time information, which can resolve occlusion problem and increase the performance. Third, we assume having 2D

doi:10.6342/NTU202100293

joints beforehand, making our model not really end-to-end. The last one is how to handle the domain gaps between real data and synthetic data. We leave these problem as our future works.

doi:10.6342/NTU202100293

Reference

[1] M. Andriluka, S. Roth, and B. Schiele, "Pictorial structures revisited: People detection and articulated pose estimation," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 1014–1021. vi, 4

[2] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*. Springer, 2016, pp. 483–499. vi, 5

[3] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4724–4732. vi, 5

[4] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2640–2649. vi, vii, 1, 6, 7, 26, 27, 31

[5] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM transactions on graphics (TOG)*, vol. 34, no. 6, pp. 1–16, 2015. vi, 2, 7, 9, 10

[6] K. Wang, J. Xie, G. Zhang, L. Liu, and J. Yang, "Sequential 3d human pose and shape estimation from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7275–7284. ix, 8, 16, 29

doi:10.6342/NTU202100293

[7] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1325–1339, 2013. ix, 29

[8] A. Gupta, J. Martinez, J. J. Little, and R. J. Woodham, "3d pose from motion for cross-view action recognition via non-linear circulant temporal encoding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2601–2608. 1, 6

[9] H.-J. Lee and Z. Chen, "Determination of 3d human body postures from a single view," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 2, pp. 148–168, 1985. 1, 6

[10] F. Moreno-Noguer, "3d human pose estimation from a single image via distance matrix regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2823–2832. 1, 6

[11] Y. Cheng, B. Yang, B. Wang, and R. T. Tan, "3d human pose estimation using spatio-temporal networks with explicit occlusion training," *arXiv preprint arXiv:2004.11822*, 2020. 1, 6

[12] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it smpl: Automatic estimation of 3d human pose and shape from a single image," in *European Conference on Computer Vision*. Springer, 2016, pp. 561–578. 2, 7, 8

[13] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7122–7131. 2, 7, 29

[14] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis, "Learning to reconstruct 3d human pose and shape via model-fitting in the loop," in *Pro-*

doi:10.6342/NTU202100293

ceedings of the IEEE International Conference on Computer Vision, 2019, pp. 2252–2261. 2, 8, 29

[15] G. Georgakis, R. Li, S. Karanam, T. Chen, J. Kosecka, and Z. Wu, "Hierarchical kinematic human mesh recovery," *arXiv preprint arXiv:2003.04232*, 2020. 2, 8, 15, 29

[16] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7103–7112. 3, 6

[17] S. Johnson and M. Everingham, "Learning effective human pose estimation from inaccurate annotation," in *CVPR 2011*. IEEE, 2011, pp. 1465–1472. 4

[18] B. Sapp, C. Jordan, and B. Taskar, "Adaptive pose priors for pictorial structures," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 422–429. 4

[19] D. Ramanan, D. A. Forsyth, and A. Zisserman, "Strike a pose: Tracking people by finding stylized poses," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 271–278. 4

[20] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on computers*, vol. 100, no. 1, pp. 67–92, 1973. 4

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 4

doi:10.6342/NTU202100293

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. 4, 5

[23] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1653–1660. 5

[24] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 648–656. 5

[25] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4929–4937. 6

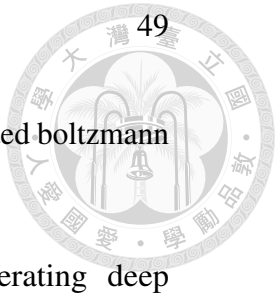
[26] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *European Conference on Computer Vision*. Springer, 2016, pp. 34–50. 6

[27] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 6

[28] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, "Towards accurate multi-person pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4903–4911. 6

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969. 6

doi:10.6342/NTU202100293



[30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010. 6

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015. 6

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014. 6

[33] H. Wu and B. Xiao, "3d human pose estimation via explicit compositional depth maps." in *AAAI*, 2020, pp. 12 378–12 385. 6

[34] B. Wandt and B. Rosenhahn, "Repnet: Weakly supervised training of an adversarial reprojection network for 3d human pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 7782–7791. 6

[35] K. Iskakov, E. Burkov, V. Lempitsky, and Y. Malkov, "Learnable triangulation of human pose," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7718–7727. 7

[36] H. Qiu, C. Wang, J. Wang, N. Wang, and W. Zeng, "Cross view fusion for 3d human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4342–4351. 7

[37] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," in *ACM SIGGRAPH 2005 Papers*, 2005, pp. 408–416. 7

[38] H. Ci, C. Wang, X. Ma, and Y. Wang, "Optimizing network structure for 3d human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2262–2271. 8

doi:10.6342/NTU202100293

[39] K. Guo, F. Xu, Y. Wang, Y. Liu, and Q. Dai, "Robust non-rigid motion tracking and surface reconstruction using l0 regularization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3083–3091. 8

[40] T. Yu, K. Guo, F. Xu, Y. Dong, Z. Su, J. Zhao, J. Li, Q. Dai, and Y. Liu, "Bodyfusion: Real-time capture of human motion and surface geometry using a single depth camera," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 910–919. 8

[41] T. Yu, Z. Zheng, K. Guo, J. Zhao, Q. Dai, H. Li, G. Pons-Moll, and Y. Liu, "Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7287–7296. 8

[42] Z. Zheng, T. Yu, H. Li, K. Guo, Q. Dai, L. Fang, and Y. Liu, "Hybridfusion: Real-time performance capture using a single depth sensor and sparse imus," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 384–400. 8

[43] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li, "Dense human body correspondences using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1544–1553. 8

[44] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660. 8, 11

[45] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108. 8, 11

doi:10.6342/NTU202100293

[46] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *CVPR*, 2017. 16

[47] Carnegie-mellon mocap database. [Online]. Available: http://mocap.cs.cmu.edu/ 16

[48] K. M. Robinette, S. Blackwell, H. Daanen, M. Boehmer, and S. Fleming, "Civilian american and european surface anthropometry resource (caesar), final report. volume 1. summary," SYTRONICS INC DAYTON OH, Tech. Rep., 2002. 16

[49] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018. 22

[50] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*, April 2020. 22

[51] K. He, X. Zhang, S. Ren, and J. Sun, "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. 24

[52] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li, "Dense human body correspondences using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 29

[53] "opencv," https://opencv.org/, accessed: 2020-11-15. 35

doi:10.6342/NTU202100293