VideoMuscle

Estimating human musculoskeletal model using neural networks from monocular video

Wei-Tse Yang



VideoMuscle

Estimating human musculoskeletal model using neural networks from monocular video

by

Wei-Tse Yang

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday September 8, 2021 at 15:00.

Student number: 5033292
Project duration: October 13, 2020 – September 8, 2021
Thesis committee: Dr. J.C. van Gemert, TU Delft, Supervisor & Chair
Dr. C. Lofi, TU Delft, Committee Member

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

The report “VideoMuscle: Estimating human musculoskeletal model using neural networks from monocular video” presents the results of my master’s thesis project at Delft University of Technology. The research was conducted under the supervision of Dr. J.C. van Gemert and Marian Bittner at TU Delft Computer Vision Lab. The document has two parts. First, scientific research exhibits the first deep learning method to infer the human skeleton system of the musculoskeletal model. Secondly, supplemental material provides the background of deep learning and relevant materials to the scientific paper.

I was inspired by applications of artificial intelligence and deep learning during my bachelor’s in biomedical engineering. My enthusiasm for deep learning and background in biomedical engineering motivated me to choose this research topic as my graduation project. During my research, I enjoyed exploring various ideas to solve challenging questions. I believed the experience would greatly help my future career.

I want to thank Jan and Marian for their tremendous support and valuable feedback through my thesis. Furthermore, I would like to acknowledge Lisa for helping me set up OpenSim’s pipeline at the beginning of my research and also appreciate Dr. C. Lofi for participating in my defense. In the end, I want to thank my family and my friends for supporting and encouraging me to pursue my career in the Netherlands.

Wei-Tse Yang
Delft, August 2021

Contents

1 Scientific Article **1**

2 The Basics of Deep Learning **19**
2.1 Deep Learning . . . 19
2.2 Convolutional Neural Networks . . . 19
2.2.1 Convolution . . . 19
2.2.2 Batch Normalization . . . 20
2.2.3 Activation function . . . 20
2.2.4 Pooling . . . 20
2.3 Sequence Models . . . 20
2.3.1 Recurrent Neural Networks . . . 21
2.3.2 Gated Recurrent Units . . . 21
2.3.3 Long Short-Term Memory . . . 21
2.3.4 Temporal Convolutional Networks . . . 22
2.3.5 3D Convolutional Neural Networks . . . 23
2.3.6 Transformer . . . 23

3 3D Human Pose Estimation via Monocular View **25**
3.1 Problem Description . . . 25
3.2 Metrics . . . 25
3.3 Deep Learning Approaches . . . 26
3.3.1 CNNs-based keypoint estimation . . . 26
3.3.2 Lifting . . . 27
3.3.3 Body Shape Estimation . . . 27
3.4 Datasets . . . 28

4 OpenSim **31**
4.1 Workflows . . . 31
4.2 Musculoskeletal Model . . . 31
4.2.1 Body . . . 32
4.2.2 Joint . . . 32
4.2.3 Kinematic Constraints . . . 32
4.2.4 Marker . . . 32
4.3 Body Scaling . . . 32
4.4 Inverse Kinematics . . . 33

1

Scientific Article

VideoMuscle : Estimating human musculoskeletal model using neural networks from monocular video

Wei-Tse Yang
Computer Vision Lab
Delft University of Technology

Marian Bittner
Computer Vision Lab
Delft University of Technology

J.C. van Gemert
Computer Vision Lab
Delft University of Technology

Abstract

We present the first deep learning approach to estimate the human skeletal system of the musculoskeletal model from monocular video. The current practice of musculoskeletal modeling relies on a motion capture system and OpenSim. The data is recorded in a restricted environment, and OpenSim workflow for musculoskeletal modeling is costly. Our research aims to solve the problem by inferring the musculoskeletal model from monocular video. Thus, our study can be applied to an arbitrary environment. We propose a spatio-temporal model to regress body scales, joint angels and one rotation matrix and exploit these three components to form a musculoskeletal model. In addition, we merge the spatial transformation of the musculoskeletal model into neural networks as an end-to-end layer to measure joint positions and marker positions. Our research demonstrates that deep learning methods can directly estimate the skeletal system of the musculoskeletal model from video effectively and efficiently. Furthermore, our proposed method has more accurate predictions on body scales and joint angles than the combination of current human pose estimation methods with OpenSim.

1. Introduction

Musculoskeletal modeling plays the main role in understanding human motion and muscle usage. A musculoskeletal model recreates skeletal and muscular systems and can simulate muscle force and joint angle during dynamic movement. The model has been applied to athlete training to prevent injury risk [7, 41], designs for rehabilitation device [54], surgical planning [19] and human activity analysis with an illness [39]. In this paper, we present the first deep learning approach to infer the skeletal system of the musculoskeletal model from monocular video.

Current practice to establish a musculoskeletal model relies on a motion capture system and OpenSim [11, 47]. OpenSim is an open-source software to help researchers develop models and simulate joint reaction and muscle force. However, the current OpenSim workflow still has some drawbacks. It is difficult to apply in outdoor or arbitrary environments because the musculoskeletal model is built onto the motion capture system. Also, it is challenging to place markers at the exact location on every subject. OpenSim requires researchers to manually adjust markers on the musculoskeletal model and iteratively check the performance until it is acceptable. The iterative adjustment makes OpenSim workflow slow and costly.

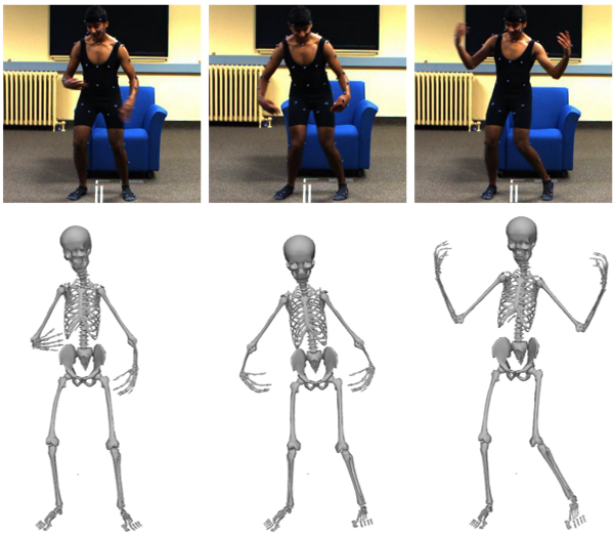

Figure 1. Qualitative examples of scaled musculoskeletal models in our dataset. Images on the top row are their corresponding frames.

Compared with motion capture data, images and videos of people performing tasks are easy to access and record. They are used to form multiple large datasets in deep learning community [12, 25] and benefit several research fields in computer vision [6, 50]. Moreover, 3D human pose estimation research has shown that neural networks can localize anatomical keypoints in 3D from images and

videos [57, 59]. Although the research has not yet estimated a human skeleton that is more complex than keypoints, the research shows the ability to learn the representation of a human pose. Therefore, we challenge OpenSim workflow and investigate how to predict the skeleton system of the musculoskeletal model from monocular video.

Our research builds a dataset of full-body musculoskeletal models for deep learning. We follow the OpenSim workflow and simulate ground-truth musculoskeletal models. Instead of using real markers as the standard approach, we exploit human meshes of Skinned Multi-Person Linear (SMPL) models [27] in Archive of Motion Capture as Surface Shapes (AMASS) [30]. We choose vertices of 3D human meshes as markers. We call them virtual markers. Virtual markers allow us to build an automatic pipeline to simulate in OpenSim without manual adjustment. Our dataset contains virtual marker data, corresponding videos, scaled musculoskeletal models and joint angles during the dynamic movement.

We propose a two-stage method to utilize both spatial and temporal information. To form a musculoskeletal model, we regress body scales, joint angles and one rotation matrix. Our method first predicts these three components per frame with a spatial model consisting of pre-trained convolutional neural networks (CNNs) and re-predict the components at the target frame with a temporal model. Moreover, we implement the spatial transformation of the musculoskeletal model as an end-to-end layer and use the operation to estimate marker positions. In our research, we focus on predicting the skeletal system of the full-body except for the hands.

To compare our proposed method, we also propose a baseline method by combining current human pose estimation research and OpenSim. We retrain the human pose estimation method to predict poses and marker positions in 3D. During the inference, we predict a sequence of marker positions from a video. Then, we utilize the sequence to generate the predicted musculoskeletal model in OpenSim.

We summarize our contributions as follows. First, we establish the first dataset of the musculoskeletal model for deep learning with an automatic pipeline. Secondly, we implement the spatial transformation of the musculoskeletal model as an end-to-end layer and make the transformation supported by the mini-batch computation. Third, we present an effective and efficient approach to infer the musculoskeletal model from monocular video. Last, to the best of our knowledge, our research is the first work that directly estimates the skeleton system of the musculoskeletal model from video.

2. Related work

OpenSim workflow. The workflow from data collection to the simulation of joint angle in OpenSim contains three steps [53]. First, retroreflective markers are placed on a subject's body and tracked by at least two infrared cameras. Secondly, the subject's calibration pose (T-pose) is recorded for body scaling, where OpenSim scales the template musculoskeletal model to the subject's body size. Finally, the subject performs the target motion, and OpenSim utilizes the sequence of marker data and the scaled model to simulate joint angles by inverse kinematics. Our research follows the OpenSim workflow to build our dataset.

Joint motion. Current studies of joint motion focus on one part of bodies. For example, Seth *et al*. [48] creates a scapulothoracic joint model with four degrees of freedom to express the activity of the scapula. Moreover, subjects are required to equip devices to gain the input of methods, such as markers, electromyography [34, 52] and inertial measurement units (IMUs) [45]. Unlike current research, we predict the full-body joint motion except for hand motions and only require one camera during the inference.

3D human pose estimation via monocular view is a task to localize anatomical keypoints from video and image with a single camera. We describe current research in deep learning into three categories: CNNs-based keypoint estimation, body shape estimation and lifting.

First, methods of CNNs-based keypoint estimation build onto pretrained convolutional neural networks to directly regress 3D joint positions. Some researchers are inspired by 2D pose estimation to represent 3D joints with heatmaps [32, 35]. The heatmap representation is usually in 2.5D, which contains two dimensions in image space and one dimension in depth. For instance, Pavlakos *et al*. [37] use stacked hourglass models to regress 2.5D heatmaps with the size of 64x64x64 per keypoint. A drawback of heatmap approaches is that the heatmap representation requires higher resolution to gain enough accuracy, which is costly in 3D poses [44]. Recently, research has shown soft-argmax allows heatmap to encode positions in lower resolution [29, 44, 49]. Moreover, Sarandi *et al*. [44] combine soft-argmax and CNNs to predict joint positions in a metric scale. The benefit of the prediction in metric space is that the model can infer poses even if joints are outside the image. In our research, we exploit the metric-scale heatmap as the spatial model of our baseline.

Secondly, the research of body shape estimation aims to predict not only joint positions but also body shapes [18, 21, 22, 33, 46, 51, 58]. These methods directly regress SMPL parameters from input images. SMPL model is the 3D human model and takes camera parameters, body coefficients and joint angles as the input. The output of the SMPL model is a 3D triangular mesh of a human. The methods exhibit that CNNs can directly infer human body shapes and joint angles. Our research follows the approaches to directly regress body scales and joint angles in a musculoskeletal model.

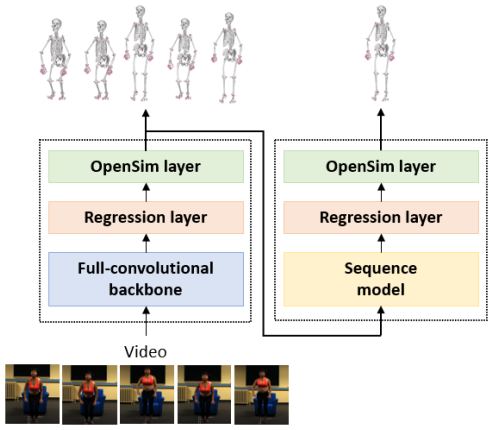

Figure 2. The proposed method. The method takes a video as a input and first predicts a musculoskeletal model per frame. Then, we exploit temporal information with a sequence model to infer a musculoskeletal model at a target frame.

Last, lifting approaches utilize pretrained 2D pose detectors and design another neural network to lift 2D poses to 3D [8, 9]. Some methods lift a pose per frame [23, 31]. Recently, the research combines temporal information with lifting to alleviate occlusion and missing detection [24, 26, 38]. These methods apply sequence models in deep learning and predict a 3D pose from a sequence of 2D poses. Lifting approaches benefit from 2D detectors trained by the large dataset of 2D human poses. The input of lifting methods is highly accurate 2D poses. Our research is difficult to apply a lifting method for the baseline because there is a lack of pretrained models and large datasets for marker positions. However, we are inspired by lifting approaches. Both the proposed method and the baseline exploit sequence models and refine predictions in 3D to 3D.

3. Method

Our proposed method takes a video as input. First, the method exploits spatial information per frame with a pretrained full-convolutional backbone and infers body scales, joint angles and a rotation matrix from the pelvis to the ground to form the musculoskeletal model. Then, the model exploits temporal relation with a sequence model and repredicts a musculoskeletal model at a target frame. We visualize the proposed method in Figure 2.

This section starts with musculoskeletal models and their kinematic transform in OpenSim. Later, we describe how to merge the kinematic transform into our proposed method as an OpenSim layer.

3.1. Musculoskeletal model

We create a full-body musculoskeletal model by merging existing models of upper limbs and lower limbs [1, 2, 10, 17, 56], thoracolumbar spine [3, 4, 5] and wrists [14, 36]. The full-body model contains all bones in a skeletal system from the head to feet and from upper arms to fingers. We do not model every degree of freedom between vertebrae because of expensive computation cost and a request for at least three markers to measure the motions of one vertebra. We separate the spine from the fifth lumbar to the first cervical vertebra into nine segments. A skeleton of a musculoskeletal model in OpenSim contains bodies, joints, kinematic constraints and markers.

Body. A body represents part of a human skeletal model as a rigid body and shares the same kinematic transform, such as a pelvis. Each body possesses one parent body and one joint. The body in OpenSim has body scales $\beta \in R^3$, which are represented by three scalars corresponding to the x,y and z axes. The body scales β are ratios to the body size of the template musculoskeletal model.

Joint. A joint defines motions between a body and its parent body. The joint has at most 3 degrees of freedom for rotation with the axis-angle representation. Each rotation represents one motion in human anatomy, like the elbow flexion. A value range of a motion is bounded in the musculoskeletal model since realistic human motion does not have complete freedom in 360 degrees. For instance, the value range of hip flexion is in $[-120°, 180°]$. The joint also defines two transformations from the joint to the body and from the joint to the parent with predefined translation T and orientation O at Euler angle representation.

Kinematic constraints. Kinematic constraints express the limitations of human motions. For example, an elbow would restrict the activities of an acromioclavicular joint. In OpenSim, the elbow is an independent joint, and the acromioclavicular joint is a dependent joint. Our musculoskeletal model exploits coordinate coupler constrain by defining a cubic spline function, where the dependent joint angle is the function output of the independent angle.

Marker. Markers represent markers placed on a subject body under a motion capture system. The marker can be represented by a point with a vector of $\vec{d} \in R^3$ from the center of its anchor body.

3.2. Kinematic transform

Kinematic transform is a process to calculate the spatial transformation of a musculoskeletal model. It is utilized to calculate ground positions of markers, joints and bodies. Bodies in a musculoskeletal model connect to their parent bodies and thus form a tree structure. Kinematic transform measures the spatial transform by traversing from the root (pelvis) to leaf nodes (hands and feet) in the level order.

First, we explain how OpenSim represents the transformation between a joint and a body. Mathematically, suppose that the translation and the orientation from a joint to its parent are $T_{parent} \in R^3$ and $O_{parent} \in R^3$ respectively.

$F(O_a)$ denotes a converting from Euler angle representation to a 3x3 rotation matrix. The transformation from a joint to its parent $R_{parent \leftarrow joint}$ is a 4x4 matrix and comprises $F(O_a)$ in the rotation part and T_{parent} in the translation part. We describe the matrix in Equation 1. We can get $R_{child \leftarrow joint}$ in the same way with T_{child} and O_{child}.

$$R_{a \leftarrow joint}(O_a, T_a) = \begin{bmatrix} F(O_a) & T_a \\ 0 & 1 \end{bmatrix}_{4x4} \quad (1)$$

To calculate the transformation from a body to its parent $R_{parent \leftarrow child}$, we first transform the body to the joint with the inverse of $R_{child \leftarrow joint}$ and multiply the joint motion matrix R_{motion}. R_{motion} describes the spatial transformation of motions. Then, we transform the joint to its parent with $R_{parent \leftarrow joint}$. We describe the whole process in Equation 2.

$$R_{parent \leftarrow child} = R_{parent \leftarrow joint}\, R_{motion}\, R^{-1}_{child \leftarrow joint} \quad (2)$$

The joint motion matrix R_{motion} represents the spatial transformation of motions. Mathematically, assume that a joint has three degrees of freedom and each degree of freedom is expressed with the axis-angle representation. We would have three predefined axes A_1, A_2, A_3 and joint angles $\theta_1, \theta_2, \theta_3$, where $A \in R^3$ and $\theta \in R^1$. R_{motion} is measured in Equation 3. Each degree of freedom is converted from an axis-angle representation to a rotation matrix, and the overall transformation is the product of these three rotation matrices. In each rotation matrix, the axis will be first projected to the previous rotation space. For instance, A_2 is transformed with $R1$ in Equation 5, where $G(A, \theta)$ is the standard function converting a axis-angle representation to a 3x3 transformation matrix.

$$R_{motion} = \begin{bmatrix} R_3 R_2 R_1 & 0 \\ 0 & 1 \end{bmatrix}_{4x4} \quad (3)$$

$$R_1 = G(A_1, \theta_1) \quad (4)$$

$$R_2 = G(R_1 A_2, \theta_2) \quad (5)$$

$$R_3 = G(R_2 R_1 A_3, \theta_3) \quad (6)$$

Finally, given a point with a vector of $\vec{d}$ from its anchored body, we can calculate its ground position P with all transformations on the path from its body to the ground in Equation 7. We exploit this operation to measure marker positions and joint positions.

$$P = \prod_{parent, child \in path} R_{parent \leftarrow child} \begin{bmatrix} \vec{d} \\ 1 \end{bmatrix}_{4x1} \quad (7)$$

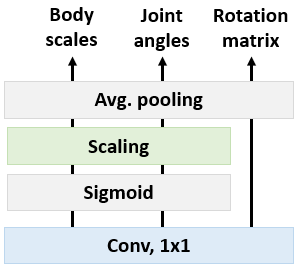

Figure 3. Regression layer. We regress body scales, joint angles and one rotation matrix. "Conv, 1x1" stands for 2d convolution with a 1x1 kernel. "Scaling" is a layer to convert body scale and joint angle to their value range with scaling and translation. "Avg. pooling" is the average pooling.

3.3. Model architecture

Regression layer. We design a module to regress three components: body scales, joint angles and one rotation matrix from the pelvis to the ground. We call it the regression layer. The architecture of the regression layer is shown in Figure 3. First, we exploit one convolutional layer with the 1x1 kernel to change the channel size of the full-convolutional backbone to a channel size of 126, which is the number of values in the three components. We explicitly explain the number of values in A.3. Secondly, each component is passed in a different branch. For the rotation matrix, we follow Zhou *et al*. [60] by predicting a 6D vector and converting it to a 3x3 rotation matrix. For joint angles and body scales, we first map features to [0,1] using a Sigmoid function and then scale and translate them to each value range. Finally, the spatial dimension of the three components is down-sampled to the size of 1x1 with the average pooling layer.

OpenSim layer. OpenSim layer is an end-to-end layer without other learnable parameters to perform the kinematic transform of a musculoskeletal model. Given body scales β, joint angles θ and rotation matrix $R_{ground \leftarrow pelvis}$, we exploit the OpenSim layer to calculate marker positions and joint positions, which are used to supervise the networks. We introduce how the prediction influences the kinematic transform as follows.

Suppose that the variable with the hat is the variable from the prediction, such as $\hat{x}$. The variable without the hat denotes the predefined variable in the musculoskeletal model. First, the translation part T in the transformation from the joint to the body depends on the subject's body scale. The intuition beyond this is that if the subject has longer legs, the center of the femur will be farther from the hip joint. We can update the translation part $\hat{T}$ by comparing the ratio between predicted body scales $\hat{\beta}$ and default body scales β in Equation 8, where $\odot$ is elementwise multiplication, and

$\oslash$ is elementwise division.

$$\hat{T} = T \odot (\hat{\beta} \oslash \beta) \tag{8}$$

Then, we can calculate the predicted transformation from the body to its parent body $\hat{R}_{parent \leftarrow child}$ in Equation 2 with predefined components $O_{parent}, O_{child}, A_1, A_2, A_3$ and predicted components $\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \hat{T}_{parent}, \hat{T}_{child}$.

In our proposed model, we directly infer the rotation matrix from the pelvis to the ground $R_{ground \leftarrow pelvis}$. $R_{ground \leftarrow pelvis}$ can initially be expressed in Equation 9, where I denotes the identity matrix. The rotation part of $R_{child \leftarrow joint}$ is also a 3x3 identity matrix in our musculoskeletal model. Our method aims to predict the root-relative position so the translation part can be ignored during the prediction. Moreover, in our musculoskeletal model, only joint angles of the pelvis are unbounded in $[-\infty,\infty]$. Predicting three unbounded angles to form the rotation matrix in Equation 3 will have the problem of discontinuity [60]. Thus, we directly predict the rotation matrix $R_{ground \leftarrow pelvis}$.

$$R_{ground \leftarrow pelvis} = I_{4x4}\, R_{motion}\, R^{-1}_{pelvis \leftarrow joint} \tag{9}$$

Last, a marker with a vector of $\vec{d}$ from the center of the body is also dependent on the body scales. The predicted vector of $\hat{d}$ is updated in Equation 10. The position of the predicted point is calculated in Equation 7 with $\hat{R}_{parent \leftarrow child}$ and $\hat{d}$.

$$\hat{d} = \vec{d} \odot (\hat{\beta} \oslash \beta) \tag{10}$$

Spatial model Spatial model is a network consisting of a full-convolutional backbone, a regression layer and an OpenSim layer. The network utilizes the spatial information from an image, such as background, objects around a human and the pose of a human. The spatial model predicts body scales, joint angles and one rotation matrix per frame.

Temporal model Temporal model contains a sequence model in deep learning, a regression layer and an OpenSim layer. The model takes the sequence of body scales, joint angles and rotation matrices as an input sequence. It exploits temporal relation from a sequence and refines body scales, joint angles and rotation matrix at a target frame. The target frame can be the middle frame or the last frame in a causal architecture [38].

3.4. Objectives

Both spatial and temporal models are supervised on joint positions, marker positions, body scales and joint angles. In our research, joint positions stands for positions of each joint in a musculoskeletal model. The overall objective function L can be expressed in Equation 11, where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weights of losses.

$$L = \lambda_1 L_{joint} + \lambda_2 L_{marker} + \lambda_3 L_{body} + \lambda_4 L_{angle} \tag{11}$$

We exploit the root-relative L1 loss in Equation 12 to define the loss of marker position L_{marker} and the loss of joint position L_{joint}. The predictions $\hat{y}$ and the labels y are first subtracted with each root position $\hat{y}_{root}, y_{root}$. For the loss of body scales L_{body} and joint angles L_{angle}, we utilize L1 norm.

$$l = \|(\hat{y} - \hat{y}_{root}) - (y - y_{root})\|_1 \tag{12}$$

3.5. Baseline

The baseline combines the method of human pose estimation with OpenSim. We modify the metric-scale heatmaps [44] as the spatial model to predict marker positions and SMPL keypoint positions in the metric scale. Like our proposed method, we exploit a sequence model to refine marker positions at the target frame.

We perform inference in two steps. First, the deep learning model infers marker positions with the spatial model under a calibration pose (T-pose). OpenSim utilizes the predicted marker data for body scaling, where the template musculoskeletal model is scaled to the subject's body size. Secondly, the spatio-temporal model predicts marker positions in the video. OpenSim exploits the sequence of marker positions to gain joint angles by inverse kinematics.

4. Experiments

4.1. Implementation details

Dataset. We utilize videos from BMLmovi dataset [13] and SMPL parameters from AMASS dataset [30]. BMLmovi is a motion and video dataset collected with IMUs, a motion capture system and four cameras. It contains 90 actors, and each subject performs 21 kinds of everyday actions. The dataset has around 9 hours of motion capture data and 17 hours of video data. We exploit videos from two calibrated PointGray cameras.

The AMASS dataset unifies current motion capture datasets into SMPL+H parameters [43]. SMPL+H is the variant of SMPL with additional parameters to represent hand motions. AMASS contains 15 different motion-capture datasets and around 426 hours of data with 346 subjects. We utilize SMPL+H parameters corresponding to BMLmovi dataset.

Our dataset combines BMLmovi with AMASS to simulate ground-truth musculoskeletal models in OpenSim. It contains 86 scaled musculoskeletal models with joint angles for around 2.7 hours of videos. Markers of a musculoskeletal model comprise 28 bony landmarks, 49 tracking markers and other 40 markers for hands, which are described explicitly in A.1. We use a training/validation/test split of 67/3/16 subjects.

Training Detail. We first utilize a pretrained Faster R-CNN [42] with ResNet-50 [15] architecture to extract a square bounding box of the human in videos. Then, the square bounding box is resized into 256x256 as the input image size.

For the spatial model, we choose ResNeXt-50 [55] as our convolutional backbone. We use an Adam optimizer with a weight decay [28] of 0.001 and a batch size of 64. The learning rate exponentially decays in two steps from 5×10^{-4} to 3.33×10^{-5} over 28 epochs and from 3.33×10^{-6} to 10^{-6} over 2 epochs. During the training, we apply data augmentation with scaling, rotation, translation and noise to simulate occlusions [44].

We explore three different architectures in the temporal model. We choose LSTM [16], temporal convolutional networks (TCNs) [38], lifting Transformer [24]. In LSTM, we utilize a bidirectional architecture with a hidden size of 128, 3 recurrent layers and a dropout probability of 0.1. In TCNs, we follow Pavllo *et al*. [38] to exploit 243 frames as the receptive field and make the momentum of batch normalization decayed from 0.1 to 0.001. In lifting Transformer, we use a hidden size of 256 and 8 parallel attention heads in the self-attention layer and a channel size of 512 in the convolutional layer.

All of the temporal models are trained with a sequence length of 243 frames and a batch size of 128. The learning rate exponentially decays from 10^{-3} to 5×10^{-6}. Both LSTM and the lifting Transformer are trained over 50 epochs with Adam optimizer [20]. TCNs are trained over 80 epochs with AMSGrad method [40].

For both temporal models and spatial models, we set the hyperparameters with $\lambda_1 = 1.0$, $\lambda_2 = 2.0$, $\lambda_3 = 0.1$ and $\lambda_4 = 0.06$.

For the baseline, the first stage training of CNNs is changed from 28 epochs to 25 epochs. Also, both spatial and temporal models are trained with a batch size of 256. Other settings are the same as those in the proposed methods.

4.2. Simulation results of our dataset

We examine the simulation results of our dataset by calculating the average root mean square error of bony landmarks. Bony landmarks are markers placed where bones are close to the surface, such as the elbow. In the practice of OpenSim, the root mean square error (RMS) for inverse kinematics should be less than 2 cm. The RMS of our dataset is 7.7 ± 4.1 mm, and there are 2% of marker positions over 2 cm. We present the qualitative results of our dataset in A.4.

4.3. Metrics

We introduce metrics for the evaluation of proposed methods and baseline.

MPBLPE. MPBLPE is the mean per bony landmarks position error. It is used to evaluate bony landmark positions. MPBLPE first aligns predictions and ground truth at the root position and calculates the average Euclidean distance.

RMS$_{body}$. Body scales are a vector of three scalars, so we directly evaluate the body scale factors by the root mean square error.

MAE$_{body}$. We further choose the axis along the longest dimension in each body scale. Then, we covert the scale of the axis into millimeters and calculate the mean absolute error (MAE$_{body}$). For the skull, toes, calcaneus, we use the x-axis. For the spine, lower limbs and upper limbs, we choose the y-axis. For the jaw, scapula, clavicle, we use the z-axis. Finally, for the pelvis, we use all three axes.

MAE$_{angle}$. MAE$_{angle}$ is the mean absolute error on the joint angle in degree.

MAPE$_{angle}$. We also evaluate joint angles with the mean absolute percentage error (MAPE$_{angle}$) because each joint angle has a different value range. Our MAPE$_{angle}$ is slightly different from the standard mean absolute percentage error. MAPE$_{angle}$ in Equation 13 converts errors to the percentage with the division of the absolute error to its value range. N, J, V, θ and $\hat{\theta}$ are the number of data, the number of joints, value ranges, ground-truth joint angles and the predicted joint angles respectively.

$$MAPE_{angle} = \frac{100}{N}\frac{1}{J}\sum_{n=1}^{N}\sum_{j=1}^{J}\frac{|\theta_j - \hat{\theta}_j|}{V_j} \tag{13}$$

In our research, MAE$_{angle}$ and MAPE$_{angle}$ are more important metrics than MPBLPE because joint angles rather than marker positions will be used for further applications of musculoskeletal models, such as joint reaction force simulation and muscle force analysis. Therefore, the accuracy of joint angles is more critical to produce precise force simulation than marker positions.

4.4. Comparison with the baseline method

We compare our method with the baseline by bony landmark positions, joint angles and body scales in Table 1. In the baseline, we extract the first frame of the video to predict marker positions of the T pose with the spatial model. The prediction of the T pose is used to simulate body scales in OpenSim. Therefore, all of the spatial and temporal models in the baseline share the same error on RMS$_{body}$ and MAE$_{body}$.

The results indicate that the proposed method can not outperform the baseline in all metrics. For example, our MPBLPE is 3.28 mm worse than the baseline on average. However, our method has better performance than baseline methods on joint angles and body scales. Significantly, our method reduces 37.4% of errors on MAE$_{angle}$ when we

compare the proposed method and baseline method on the Transformer architecture.

We further examine errors on body scales and joint angles. In Table 2, the results demonstrate that the baseline has a bias on upper limbs, and the proposed method has larger errors on lower limbs. In Table 3, the baseline and the proposed method have comparable performance on joint angles of the pelvis and lower limbs. However, the proposed method outperforms the baseline on the angles of shoulders, elbows and radioulnar joints.

Furthermore, we compare the running time on Windows 10 with four cores, 52 GB RAM and NVIDIA T4 GPU. We choose the lifting Transformer as the architecture in both the baseline and proposed method. For the baseline method, OpenSim is executed in parallel with four cores. Our proposed method achieves 31.96 fps and outperforms the baseline method in Table 4. Since the OpenSim layer must traverse bodies in the level order, our proposed method is slow during a batch size of 1. However, the support of mini-batch computation in OpenSim layer can alleviate the problem, and thus our method can run faster than the baseline.

4.5. Qualitative results

We visualize the prediction of musculoskeletal models from our proposed method with the Transformer architecture in Figure 4. We take the average body scales of the predicted sequence as scales of the model to visualize in OpenSim. Moreover, we present qualitative results of joint angels in A.4.

4.6. Ablation study

Losses. We examine the effect of the losses in Table 5. We train the spatial models by gradually turning on each loss. The results show that training only with L_{marker} can achieve the lowest MPBLPE, but it has the largest errors on body scales and joint angles. Moreover, although training without the loss of body scales obtains the lowest error on joint angles, the performance of body scales is worse than the baseline. Thus, we keep using all four losses in our research.

Loss type. We investigate the influence of the L1 norm and the L2 norm. We conduct two studies on spatial models and temporal models separately. First, we show that training spatial models with L1 norm performs better on joint angles and body scales in Table 6. Secondly, we utilize the prediction from the spatial model trained by the L1 norm as the input of sequence models. We examine the effect of the L1 norm and the L2 norm on the temporal models. We obtain better results on joint angles and body scales with the L1 norm in Table 7. Moreover, training with L2 norm in sequence models degrades the performance on body scales. In our research, we also explore the Huber loss, but the Huber loss does not improve the performance, which is discussed in A.2.

Temporal smoothness. We study the benefit of the usage of temporal models. In Table 1, we have shown that using temporal models can decrease errors on bony landmark positions, body scales and joint angles. We further measure the smoothness of the predicted sequence. We estimate the mean velocity error on bony landmark positions and joint angles, denoted as MVE_{BL} and MVE_{angle} respectively. The mean velocity error is calculated by the first derivative of the data. We demonstrate that using temporal models can improve the smoothness of the sequence in Table 8.

5. Limitations and Conclusion

Our research proposes the first deep learning approach to infer the skeleton system of the musculoskeletal model from monocular video. Direct inference from videos alleviates reliance on a motion capture system in current practice and allows the establishment of musculoskeletal models in arbitrary environments. Our research proposes a spatio-temporal model. The method regresses body scales, joint angles and one rotation matrix per frame with a spatial model and refines predictions with a temporal model. Although the baseline has a lower error on bony landmark positions, our proposed method achieves more accurate performance on body scales and joint angles. Joint angles are more critical than bony landmarks positions for force simulation of musculoskeletal models. As a result, our proposed model is more suitable for establishing musculoskeletal models and simulating muscle force.

Our research leaves some limitations to future work. First, our dataset has already simulated hand motions, but we focus on predicting the whole body except for hands since methods might need sub-networks to process hands images. Secondly, our research is built onto virtual data. We use virtual markers rather than real markers to prevent manual adjustment in OpenSim. Finally, we found out that SMPL parameters had more freedom than human bodies. To let our musculoskeletal model fitted with the data, we unlock six kinematic constraints and increase six degrees of freedom in joints. As a result, the performance of our method still needs to be verified when applied to the real world. Future work needs to build a dataset from real motion capture data to solve the problem.

References

[1] FRANK C. ANDERSON and MARCUS G. PANDY. A dynamic optimization solution for vertical jumping in three dimensions. *Computer Methods in Biomechanics and Biomedical Engineering*, 2(3):201–231, 1999. PMID: 11264828. 3

[2] Frank C. Anderson and Marcus G. Pandy. Dynamic Optimization of Human Walking . *Journal of Biomechanical Engineering*, 123(5):381–390, 05 2001. 3

		MPBLPE (mm)	RMS_{body}	MAE_{body} (mm)	MAE_{angle} (°)	$MAPE_{angle}$(%)
Ours	Spatial	37.78	0.085	6.07	3.58	3.89
	Spatial + LSTM	37.61	0.082	5.97	3.57	3.87
	Spatial + TCNs	38.06	0.083	5.93	3.54	3.87
	Spatial + Transformer	**36.98**	**0.082**	**5.90**	**3.54**	**3.86**
Baseline	Spatial	35.04	0.12	6.25	5.89	6.35
	Spatial + LSTM	**33.74**	-	-	5.79	6.19
	Spatial + TCNs	34.52	-	-	5.82	6.23
	Spatial + Transformer	34.00	-	-	**5.66**	**6.07**

Table 1. Comparison of bony landmarks position, body scales and joint angles between our method and baseline.

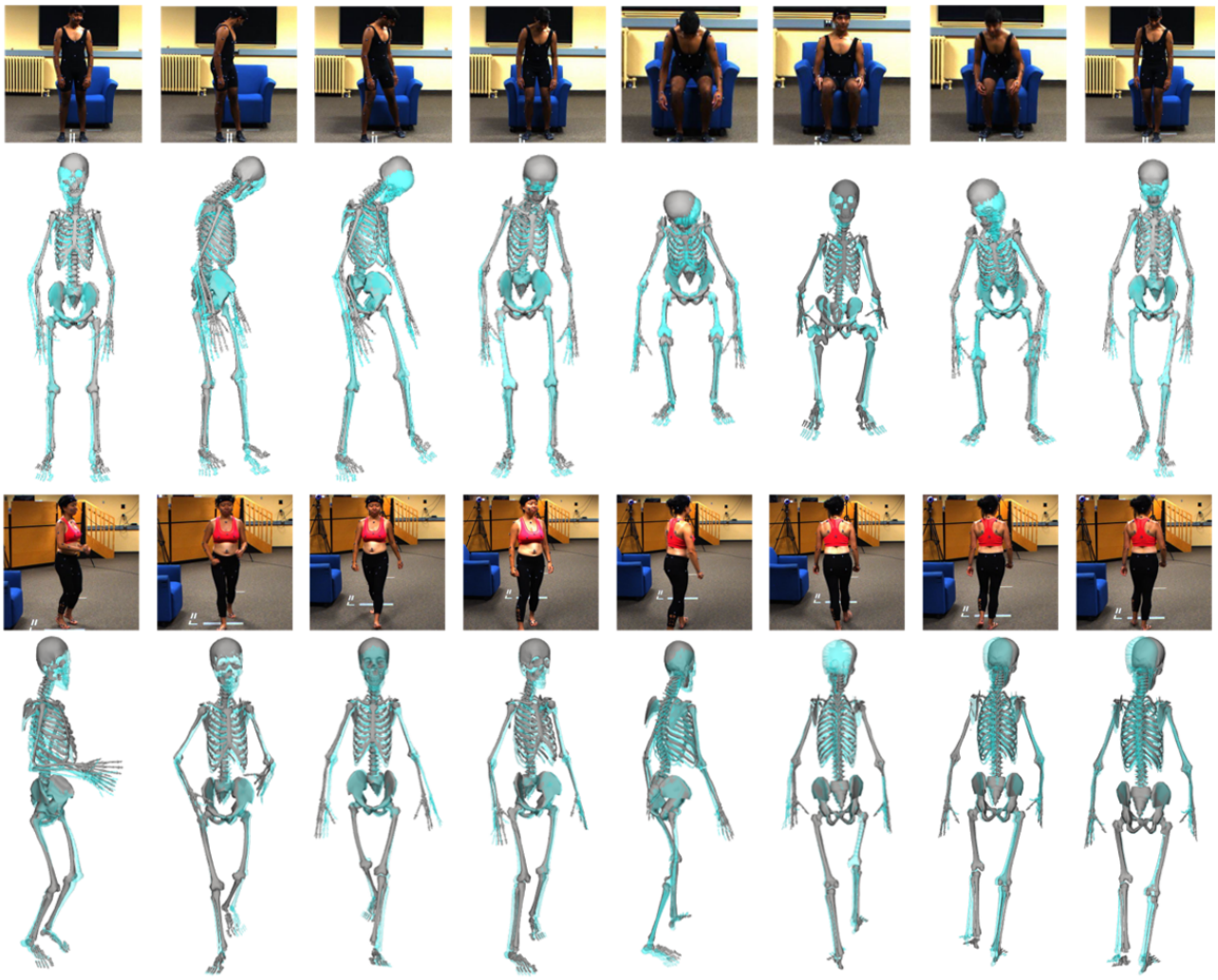

Figure 4. Qualitative results of our proposed method. We visualize the overlapping between the predicted musculoskeletal model and the ground truth. Blue human is the ground-truth musculoskeletal model. The gray human is the prediction from our proposed method.

[3] Alexander G. Bruno, Mary L. Bouxsein, and Dennis E. Anderson. Development and Validation of a Musculoskeletal Model of the Fully Articulated Thoracolumbar Spine and Rib Cage. *Journal of Biomechanical Engineering*, 137(8), 06 2015. 081003. 3

[4] Alexander G Bruno, Katelyn Burkhart, Brett Allaire, Dennis E Anderson, and Mary L Bouxsein. Spinal loading patterns from biomechanical modeling explain the high in-

RMS_{body} (MAE_{body})		
	Baseline	Ours
pelvis	**0.090 (9.58)**	0.091 (9.82)
femur	**0.073 (10.55)**	0.091 (22.21)
tibia	**0.060 (9.82)**	0.102 (35.00)
humerus	0.102 (14.55)	**0.068 (9.41)**
ulna	0.395 (24.91)	**0.075 (11.59)**
radius	0.395 (23.60)	**0.075 (10.98)**

Table 2. Comparison of body scales.

MAE_{angle} (MAPE_{angle})		
	Baseline	Ours
hip flexion	4.91 (1.64)	**4.78 (1.59)**
hip abduction	2.31 (1.28)	**1.97 (1.09)**
hip rotation	4.43 (2.46)	**4.11 (2.28)**
knee angle	**4.50 (2.81)**	4.82 (3.01)
ankle angle	5.13 (3.69)	**4.30 (3.10)**
subtalar angle	3.60 (5.14)	**3.49 (4.98)**
shoulder angle	21.43 (7.94)	**13.79 (5.11)**
shoulder elevation	10.67 (5.61)	**6.34 (3.34)**
shoulder rotation	18.91 (10.50)	**8.13 (4.52)**
elbow flexion	8.05 (4.02)	**6.44 (3.22)**
supination and pronation of radioulnar joint	23.01 (8.22)	**13.38 (4.78)**

Table 3. Comparison of joint angles.

Batch size	1	16	64	128	256
Ours	0.92	8.78	20.94	28.25	**31.96**
Baseline	7.51	8.36	8.43	**8.44**	8.35

Table 4. Comparison of time cost in fps.

	L_{marker}	$+L_{joint}$	$+L_{angle}$	$+L_{body}$
L_{marker}	√	√	√	√
L_{joint}		√	√	√
L_{angle}			√	√
L_{body}				√
MPBLPE	**35.56**	36.26	37.08	37.78
RMS_{body}	0.197	0.168	0.131	**0.085**
MAE_{body}	8.46	6.78	6.27	**6.07**
MAE_{angle}	14.58	11.53	**3.54**	3.58
MAPE_{angle}	15.09	11.78	**3.86**	3.89

Table 5. The ablation study for the effect of losses on the spatial model. The experiment starts from the training only with L_{marker} and gradually turn on each loss. For instance, $+L_{angle}$ stands for training with L_{marker}, L_{joint} and L_{angle}.

Spatial model	L1	L2
MPBLPE	37.78	**36.37**
RMS_{body}	**0.085**	0.087
MAE_{body}	**6.07**	6.25
MAE_{angle}	**3.58**	3.61
MAPE_{angle}	**3.89**	4.01

Table 6. The ablation study for loss types on the spatial model. $L1$ and $L2$ stand for L1 norm and L2 norm.

	LSTM		TCNs		Transformer	
	L1	L2	L1	L2	L1	L2
MPBLPE	**37.61**	37.71	38.06	**37.05**	36.98	**36.71**
RMS_{body}	**0.082**	0.091	**0.083**	0.090	**0.082**	0.091
MAE_{body}	**5.97**	6.74	**5.93**	6.42	**5.90**	6.42
MAE_{angle}	**3.57**	3.61	**3.54**	3.57	**3.54**	3.57
MAPE_{angle}	**3.87**	4.01	**3.87**	3.98	**3.86**	3.96

Table 7. The ablation study for loss types on sequence models. $L1$ and $L2$ are L1 norm and L2 norm.

	MVE_{BL} (mm)	MVE_{angle} (°)
Spatial model	12.60	4.18
+ LSTM	8.10	4.03
+ TCNs	8.17	3.93
+ Transformer	8.76	3.91

Table 8. The mean velocity errors for bony landmarks MVE_{BL} and joint angles MVE_{angle}.

cidence of vertebral fractures in the thoracolumbar region. *Journal of Bone and Mineral Research*, 32(6):1282–1290, 2017. 3

[5] Katelyn Burkhart, Daniel Grindle, Mary L. Bouxsein, and Dennis E. Anderson. Between-session reliability of subject-specific musculoskeletal models of the spine derived from optoelectronic motion capture data. *Journal of Biomechanics*, 112:110044, 2020. 3

[6] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019. 1

[7] Jing Chang, Wenrui Wang, Damien Chablat, and Fouad Bennis. Evaluating the risk of muscle injury in football-kicking training with opensim. In Vincent G. Duffy, editor, *Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Human Body, Motion and Behavior*, pages 100–109, Cham, 2021. Springer International Publishing. 1

[8] Yu Cheng, Bo Yang, Bo Wang, and Robby T. Tan. 3d human pose estimation using spatio-temporal networks with explicit occlusion training, 2020. 3

[9] Yu Cheng, Bo Yang, Bo Wang, Wending Yan, and Robby T. Tan. Occlusion-aware networks for 3d human pose estimation in video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3

[10] S.L. Delp, J.P. Loan, M.G. Hoy, F.E. Zajac, E.L. Topp, and J.M. Rosen. An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. *IEEE Transactions on Biomedical Engineering*, 37(8):757–767, 1990. 3

[11] Scott L. Delp, Frank C. Anderson, Allison S. Arnold, Peter Loan, Ayman Habib, Chand T. John, Eran Guendelman, and Darryl G. Thelen. Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1940–1950, 2007. 1

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[13] Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F. Troje. Movi: A large multi-purpose human motion and video dataset. *PLOS ONE*, 16(6):1–15, 06 2021. 5

[14] R. Gonzalez, T. Buchanan, and S. Delp. How muscle architecture and moment arms affect wrist flexion-extension moments. *Journal of biomechanics*, 30 7:705–12, 1997. 3

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 6

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. 6

[17] Katherine R S Holzbaur, Wendy M Murray, and Scott L Delp. A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control. *Annals of biomedical engineering*, 33(6):829—840, June 2005. 3

[18] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Regognition (CVPR)*, 2018. 2

[19] Akin Oguz Kapti. Dynamic simulation of tibialis posterior tendon transfer in the treatment of drop-foot. *Biocybernetics and Biomedical Engineering*, 34(2):132–138, 2014. 1

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 6

[21] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation, 2020. 2

[22] Nikos Kolotouros, Georgios Pavlakos, Michael J. Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop, 2019. 2

[23] Shichao Li, Lei Ke, Kevin Pratama, Yu-Wing Tai, Chi-Keung Tang, and Kwang-Ting Cheng. Cascaded deep monocular 3d human pose estimation with evolutionary training data. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. 3

[24] Wenhao Li, Hong Liu, Runwei Ding, Mengyuan Liu, and Pichao Wang. Lifting transformer for 3d human pose estimation in video, 2021. 3, 6

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 1

[26] Ruixu Liu, Ju Shen, He Wang, Chen Chen, Sen ching Cheung, and Vijayan K. Asari. Enhanced 3d human pose estimation from videos by using attention-based neural network with dilated convolutions, 2021. 3

[27] Matthew Loper, Naureen Mahmood, and Michael J. Black. Mosh: Motion and shape capture from sparse markers. *ACM Trans. Graph.*, 33(6), Nov. 2014. 2

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 6

[29] Diogo C. Luvizon, Hedi Tabia, and David Picard. Human pose regression by combining indirect part detection and contextual information, 2017. 2

[30] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, pages 5442–5451, Oct. 2019. 2, 5

[31] Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. A simple yet effective baseline for 3d human pose estimation, 2017. 3

[32] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect. *ACM Transactions on Graphics*, 36(4):1–14, Jul 2017. 2

[33] Gyeongsik Moon and Kyoung Mu Lee. Neuralannot: Neural annotator for in-the-wild expressive 3d human pose and mesh training sets, 2021. 2

[34] Marion Mundt, Arnd Koeppe, Franz Bamer, Sina David, and Bernd Markert. Artificial neural networks in motion analysis—applications of unsupervised and heuristic feature selection techniques. *Sensors*, 20(16), 2020. 2

[35] Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. 3d human pose estimation with 2d marginal heatmaps, 2018. 2

[36] OpenSim. Modification of wrist model to include all the movements of the fingers. SimTK, February. 28, 2018. [Online]. 3

[37] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G. Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose, 2017. 2

[38] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training, 2019. 3, 5, 6

[39] Lorenzo Pitto, Hans Kainz, Antoine Falisse, Mariska Wesseling, Sam Van Rossom, Hoa Hoang, Eirini Papageorgiou, Ann Hallemans, Kaat Desloovere, Guy Molenaers, Anja Van Campenhout, Friedl De Groote, and Ilse Jonkers. Simcp: A simulation platform to predict gait performance following orthopedic intervention in children with cerebral palsy. *Frontiers in Neurorobotics*, 13:54, 2019. 1

[40] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond, 2019. 6

[41] Jeffrey A. Reinbolt, Ajay Seth, and Scott L. Delp. Simulation of human movement: applications using opensim. *Procedia IUTAM*, 2:186–198, 2011. IUTAM Symposium on Human Body Dynamics. 1

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 6

[43] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Trans. Graph.*, 36(6), Nov. 2017. 5

[44] Istvan Sarandi, Timm Linder, Kai O. Arras, and Bastian Leibe. Metric-scale truncation-robust heatmaps for 3d human pose estimation. *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, Nov 2020. 2, 5, 6

[45] Thomas Seel, Jörg Raisch, and Thomas Schauer. Imu-based joint angle measurement for gait analysis. *Sensors*, 14(4):6891–6909, 2014. 2

[46] Akash Sengupta, Ignas Budvytis, and Roberto Cipolla. Synthetic training for accurate 3d human pose and shape estimation in the wild, 2020. 2

[47] Ajay Seth, Jennifer L. Hicks, Thomas K. Uchida, Ayman Habib, Christopher L. Dembia, James J. Dunne, Carmichael F. Ong, Matthew S. DeMers, Apoorva Rajagopal, Matthew Millard, Samuel R. Hamner, Edith M. Arnold, Jennifer R. Yong, Shrinidhi K. Lakshmikanth, Michael A. Sherman, Joy P. Ku, and Scott L. Delp. Opensim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLOS Computational Biology*, 14(7):1–20, 07 2018. 1

[48] Ajay Seth, Ricardo Matias, António P. Veloso, and Scott L. Delp. A biomechanical model of the scapulothoracic joint to accurately capture scapular kinematics during shoulder movements. *PLOS ONE*, 11(1):1–18, 01 2016. 2

[49] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression, 2018. 2

[50] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. 1

[51] Shashank Tripathi, Siddhant Ranade, Ambrish Tyagi, and Amit Agrawal. Posenet3d: Learning temporally consistent 3d human pose via knowledge distillation, 2020. 2

[52] Triwiyanto Triwiyanto, O. Wahyunggoro, H. A. Nugroho, and H. Herianto. Evaluating the performance of kalman filter on elbow joint angle prediction based on electromyography. *International Journal of Precision Engineering and Manufacturing*, 18:1739–1748, 2017. 2

[53] Mukul Verma, Muskan Dawar, Prashant Singh Rana, Neeru Jindal, and Harpreet Singh. Chapter 3 - an opensim guided tour in machine learning for e-health applications. In Amit Kumar Singh and Mohamed Elhoseny, editors, *Intelligent Data Security Solutions for e-Health Applications*, Intelligent Data-Centric Systems, pages 57–75. Academic Press, 2020. 2

[54] Michele Vivian, Luca Tagliapietra, Massimo Sartori, and Monica Reggiani. Dynamic simulation of robotic devices using the biomechanical simulator opensim. In Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *Intelligent Autonomous Systems 13*, pages 1639–1651, Cham, 2016. Springer International Publishing. 1

[55] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017. 6

[56] Gary T. Yamaguchi and Felix E. Zajac. A planar model of the knee joint to characterize the knee extensor mechanism. *Journal of Biomechanics*, 22(1):1–10, 1989. 3

[57] Wei Yang, Wanli Ouyang, Xiaolong Wang, Jimmy Ren, Hongsheng Li, and Xiaogang Wang. 3d human pose estimation in the wild by adversarial learning, 2018. 2

[58] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. Simpoe: Simulated character control for 3d human pose estimation, 2021. 2

[59] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N. Metaxas. Semantic graph convolutional networks for 3d human pose regression. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. 2

[60] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks, 2020. 4, 5

A. Supplementary material

A.1. Markers

We present virtual markers on our musculoskeletal model in Figure 5 and Figure 6. There are 28 bony landmarks, 49 tracking markers and other 40 markers on hands. Joints of the SMPL model are treated as tracking markers in our research.

A.2. The normalized Huber loss

Our research also exploits the normalized Huber loss as the loss of joint angles L_{angle} and body scales L_{body}. Huber loss possesses a threshold α to decide the usage of L1 loss and L2 loss, shown in Equation 15. Since each joint angle or body scale has a different value range, it is unsuitable for applying a threshold to all joint angles or body scales. Therefore, we combine the normalization with the Huber loss. We call the process the normalized Huber loss. First, we map each error from its value range V to the unified scale. The unified scale is $\frac{\alpha}{H}$ with one more hyperparameter H. Then, we calculate Huber loss and re-scale the error back to its original scale. The normalized Huber loss is described in Equation 14.

$$l = \frac{2VH}{\alpha} Huber(\frac{\alpha}{H}\frac{1}{2V}(\hat{y}-y)) \tag{14}$$

$$Huber(\hat{y}-y) = \begin{cases} \frac{1}{2\alpha}(\hat{y}-y)^2, & |\hat{y}-y| < \alpha \\ |\hat{y}-y| - 0.5\alpha, & \text{otherwise} \end{cases} \tag{15}$$

We prove α in the normalized Huber loss can be an arbitrary value, and H represents a ratio of original value ranges V as the threshold in Huber loss. By bringing Equation 14 to Equation 15, α will be canceled in Equation 16. The intuition is that we map value ranges to a unified scale $\frac{\alpha}{H}$, and thus the scale controlled by α can be an arbitrary value.

$$\begin{cases} \frac{2VH}{\alpha}\frac{1}{2\alpha}(\frac{\alpha}{2VH}(\hat{y}-y))^2 & \frac{\alpha}{2VH}|\hat{y}-y| < \alpha \\ = \frac{1}{4VH}(\hat{y}-y)^2, & \Rightarrow \frac{|\hat{y}-y|}{2V} < H \\ \frac{2VH}{\alpha}(\frac{\alpha}{2VH}|\hat{y}-y| - 0.5\alpha) & \\ = |\hat{y}-y| - VH, & \text{otherwise} \end{cases} \tag{16}$$

We show that the normalized Huber loss does not improve the performance when we compare the training of spatial models by L1 norm, L2 norm and the normalized Huber loss in Table 9. Although training with the normalized Huber loss has the lowest error on MAE_{body}, MAE_{angle} and MAPE_{angle} are slightly worse than those of L1 norm. Since the performance of joint angles is the most important in our research, we discard the normalized Huber loss and exploit the L1 norm.

	L1	L1/Huber	L2
L_{joint}	$L1$	$L1$	$L2$
L_{marker}	$L1$	$L1$	$L2$
L_{body}	$L1$	nH	$L2$
L_{angle}	$L1$	nH	$L2$
MPBLPE	37.78	36.72	36.37
RMS_{body}	0.085	0.084	0.087
MAE_{body}	6.07	6.03	6.25
MAE_{angle}	3.58	3.61	3.61
MAPE_{angle}	3.89	3.92	4.01

Table 9. Comparison of loss types on the spatial model. $L1$, nH and $L2$ are L1 norm, the normalized Huber loss and L2 norm.

A.3. Body scales

We explain how to conduct body scaling in OpenSim and how to predict body scales in the regression layer in this section.

Body scaling in OpenSim is a process to scale the template musculoskeletal model to a subject's body size. OpenSim conducts the scaling by comparing the ratio of the distances of marker pairs on the template model to the experimental marker data. For instance, body scales are a vector of (1,1,1) and the distance of the marker pair used for scaling is 0.5 m. In the experiment, the distance of the marker pair is 1 m. The new scale would be a vector of (2,2,2).

Typically, body scales of the template model are vectors of (1,1,1). Also, some bodies would use the same marker pair for body scaling since the number of markers is usually less than the number of bodies. For example, we scale the y-axis of vertebrae from the fifth lumbar to the first lumbar with a marker pair of T12 and MPSI. MPSI is the median point of RPSI and LPSI. As a result, bodies would have the same values after body scaling if they have template scales of (1,1,1) and use the same marker pairs.

We know that there are repeated values in our musculoskeletal model. To simplify our method, we only predict unique values of body scales in the regression layer and allocate values back to each body. In summary, we predict 51 unique values for 48 bodies, 69 joint angles and 6 values for a 6D vector of rotation matrix in the regression layer, where the output channel size of 1x1 convolutional layer is 126.

A.4. Qualitative results

This section presents qualitative results on joint angles and our dataset. First, we compare ground-truth angles with predicted joint angles from our proposed method with the Transformer architecture. We exhibit left knee angle in Figure 7 and elbow flexion of right hands in Figure 8. Secondly, we visualize the qualitative results of our dataset in Figure 9. The figure contains video frames from BMLmovi, poses of musculoskeletal models and corresponding scaled

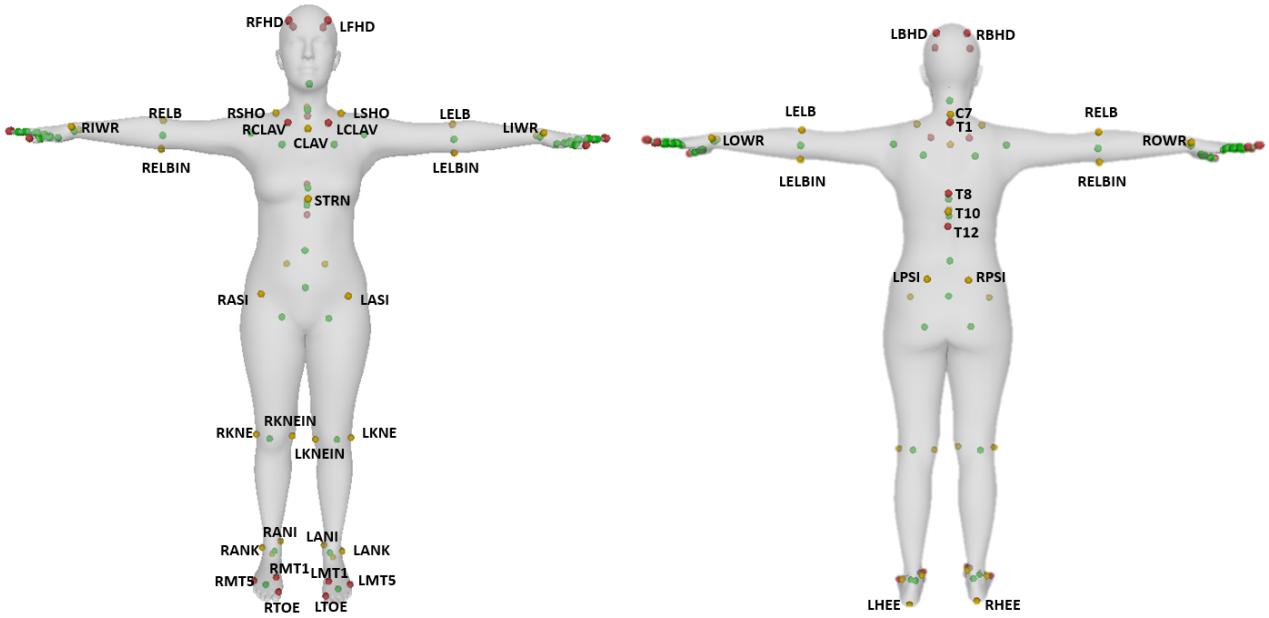

Figure 5. Virtual markers. We visualize bony landmarks as yellow markers, and red markers are tracking markers. Green markers are joints of the SMPL model. Our research treats SMPL joints as tracking markers.

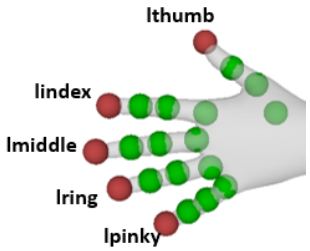

Figure 6. Virtual markers on left hand. Red markers are markers on fingertips, and green markers are joints of the SMPL model.

musculoskeletal models. The poses are defined by linking joint positions of musculoskeletal models.

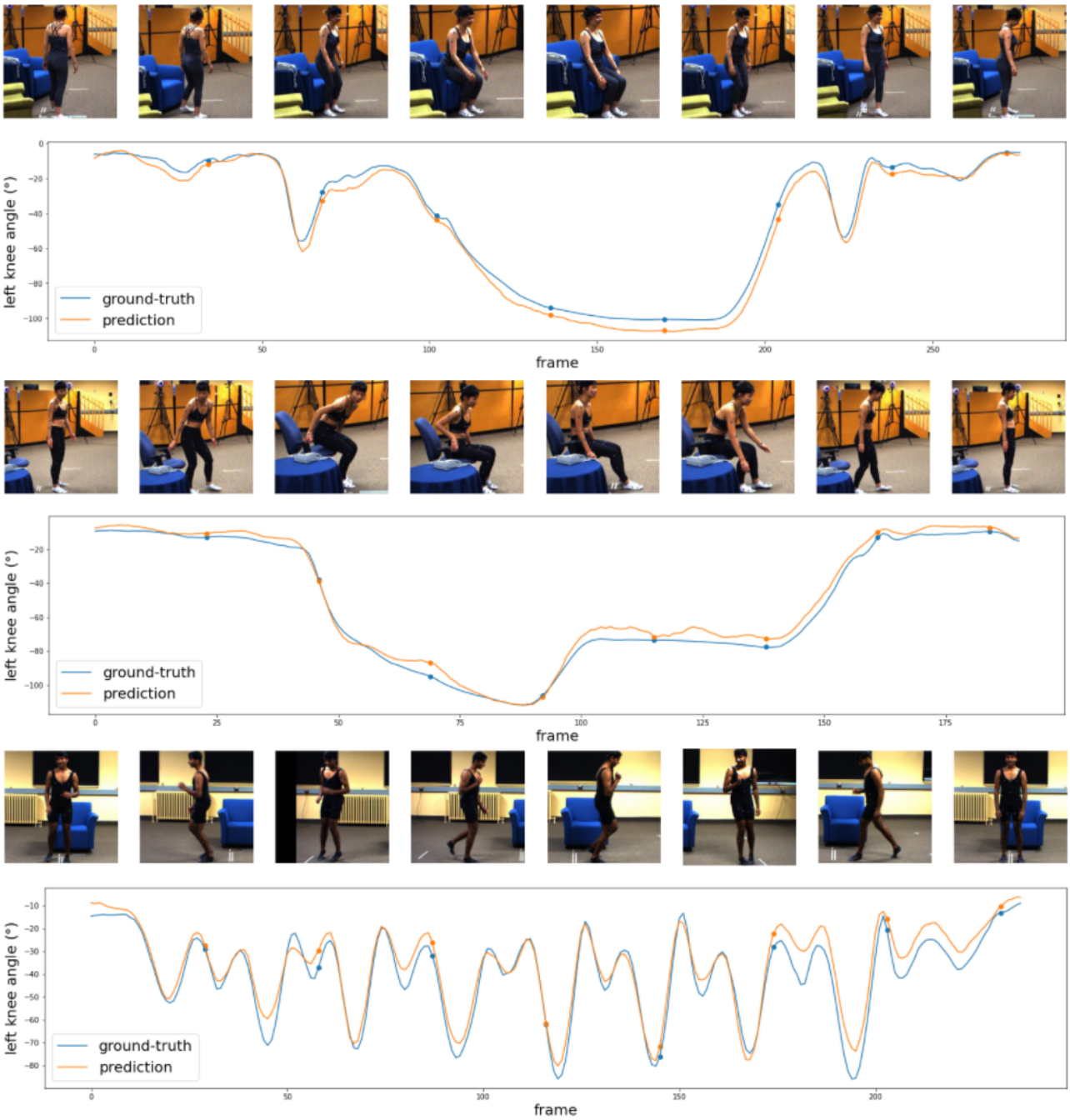

Figure 7. Qualitative results of our proposed method on left knee angle. We visualize predictions of the Transformer architecture and their ground-truth angles with their corresponding frames. In each line plot, the i-th dot represents the frame of the i-th image shown above the line plot.

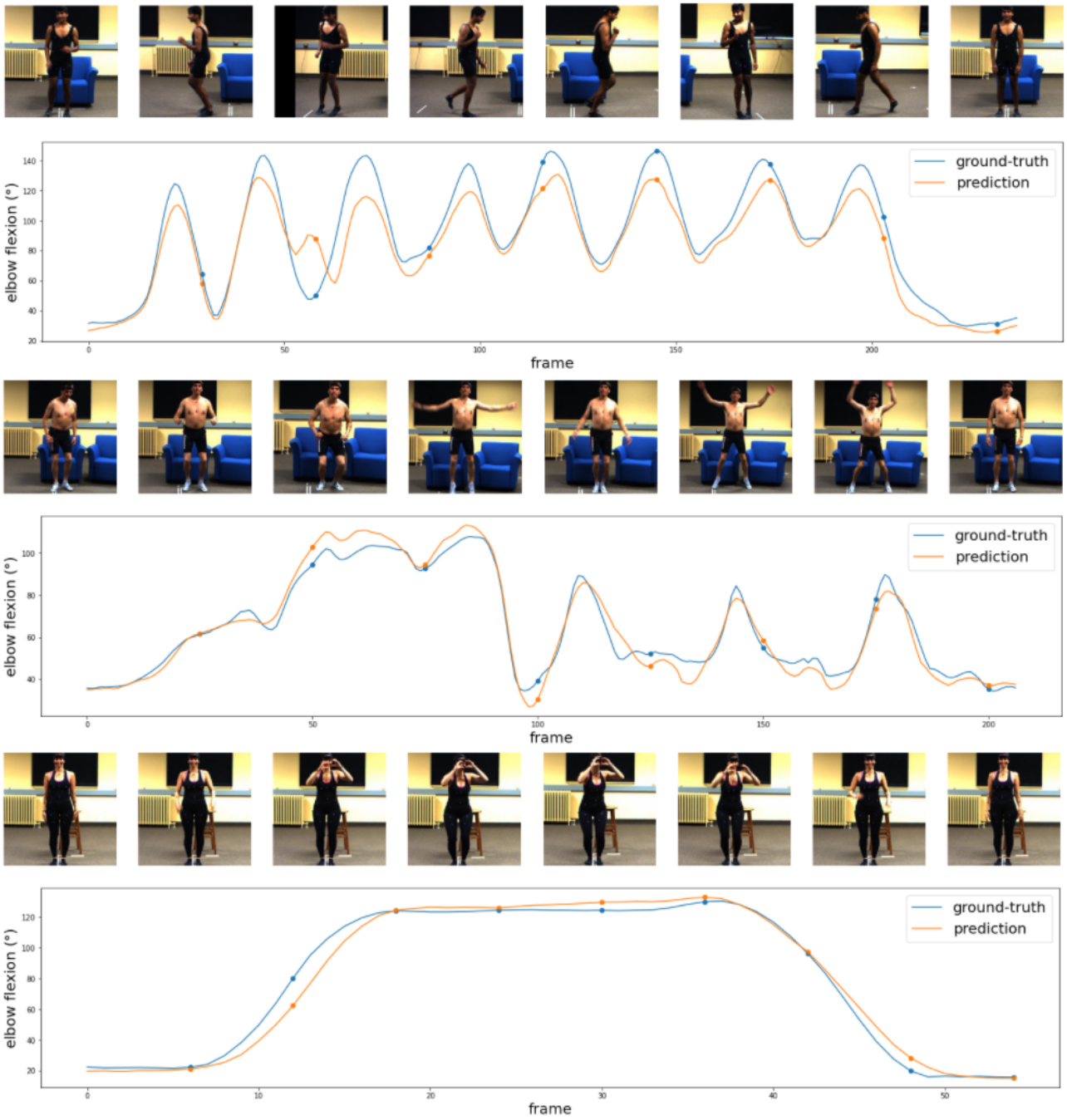

Figure 8. Qualitative results of our proposed method on elbow flexion of right hand. We visualize predictions of the Transformer architecture and their ground-truth angles with their corresponding frames. In each line plot, the i-th dot represents the frame of the i-th image shown above the line plot.



Figure 9. Qualitative results of our dataset. The first column and the fourth column are video frames in the BMLmovi dataset. The second column and the fifth column are their poses by linking joint positions of musculoskeletal models. Markers of musculoskeletal models are presented with orange dots around poses. The third column and the sixth column are scaled musculoskeletal models corresponding with video frames.

2

The Basics of Deep Learning

2.1. Deep Learning

Deep learning is a sub-field of machine learning. Conventional machine learning approaches perform tasks with handcrafted features, which are manually estimated from raw data. However, deep learning methods perform feature extraction and prediction automatically in a single model. They utilize several neural layers to construct a neural network. The neural network can extract low-level features from raw data and progressively converts them to high-level features relevant to the task. For instance, in image classification, the network would first extract image features, such as edges and textures. Then, the network learns high-level representation for each class.

Recently, Deep learning has shown promising results in multiple areas, such as image classification [36], object detection [1], human pose estimation [2] and machine translation [39].

2.2. Convolutional Neural Networks

Convolutional neural networks (CNNs) have been widely used in computer vision of deep learning. The networks comprise multiple stacks of the four key components: convolutional layer, batch normalization, activation functions and pooling layer. First, the convolutional layer slides over images to extract features. Secondly, batch normalization normalizes feature maps. Third, activation functions make the weighted sum from convolutional operation nonlinear. Finally, the pooling layer reduces the spatial dimension and filters out redundant features.

2.2.1. Convolution

The main idea of the convolutional layer uses the small window to slide over feature maps in Figure 2.1. The small window is called either the filter or the kernel. The filter is the standard technology in image processing. It is assigned with different weights to detect specific features, such as edge detection.

Convolution has two characteristics: local connection and weights sharing. Compared with the fully connected network, the two characteristics allow CNNs to reduce the number of parameters. The convolutional operation only computes nearby elements of features maps without connecting with all elements in the previous layer. In addition, it exploits the kernel with the same weights to slide over images. These two characteristics allow the network to be formed in deeper architecture.

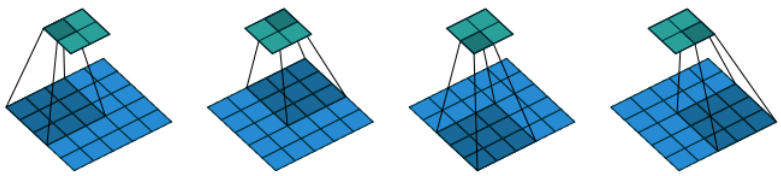

Figure 2.1: Convolutional operation with a 3x3 kernel [8].

2.2.2. Batch Normalization

Batch normalization [12] is widely used in CNNs to accelerate training and reduce internal covariance shift. It first normalizes feature maps x and scales and translates them to a new scale with two learnable parameters γ,β. We show the formula in Equation 2.1, where E[x], Var[x] and ϵ are expected values, variance and a hyperparameter to prevent from dividing by zero respectively.

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \times \gamma + \beta \tag{2.1}$$

2.2.3. Activation function

The convolutional layer produces the weighted sum of the kernel and the features, so the operation is linear. However, nonlinear models are required in deep learning methods to possess enough model complexity for representation learning. As a result, activation functions are added after the convolutional layer and batch normalization to make CNNs nonlinear. We introduce common activation functions as follows.

Rectified Linear Unit (ReLU) [28] in Equation 2.2 is the most widely used activation function in CNNs. The function maximizes the input and zero.

$$ReLU(x) = max(x, 0) \tag{2.2}$$

Sigmoid in Equation 2.4 is another activation function to map the input to [0,1].

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

Softmax is an activation function to make the sum of all input elements equal to 1. It is widely used in multiclass classification.

$$Softmax(x_i) = \frac{x_i}{\sum_{j=1}^{J} x_j} \tag{2.4}$$

2.2.4. Pooling

The pooling layer reduces the spatial dimension by extracting the average/maximum value in a small window. The pooling layer brings several advantages to CNNs. First, the pooling filters out redundant features and allow exhibited features forward pass through the network. Secondly, CNNs can reduce the memory cost since the spatial dimension of feature maps is down-sampled through the forward pass. Last, reducing the spatial dimension of features can increase the receptive field of CNNs.

Typically, there are two types of pooling layers: the average pooling and the maximum pooling. Examples are shown in Figure 2.2. The former extracts the average value, and the latter chooses the maximum value.



(a) The average pooling

(b) The maximum pooling

Figure 2.2: The pooling layer [8].

2.3. Sequence Models

Sequence models are deep learning approaches for time-series data. This section starts with LSTM-based recurrent neural networks, which exploit the hidden state to remember the relation in the sequence. Then, we introduce temporal convolutional networks and 3D CNNs. Convolution-based methods do not use hidden states and directly apply convolutional operations. Last, we explain the self-attention mechanism and Transformer.

2.3.1. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) in Figure 2.3 use the hidden state to model the relation in the sequence. The networks maintain the hidden state H and let the hidden state of the previous step H_{t-1} be used in the current stage t. RNNs use the same weights to traverse the sequence. Ideally, the networks can process any sequence length.

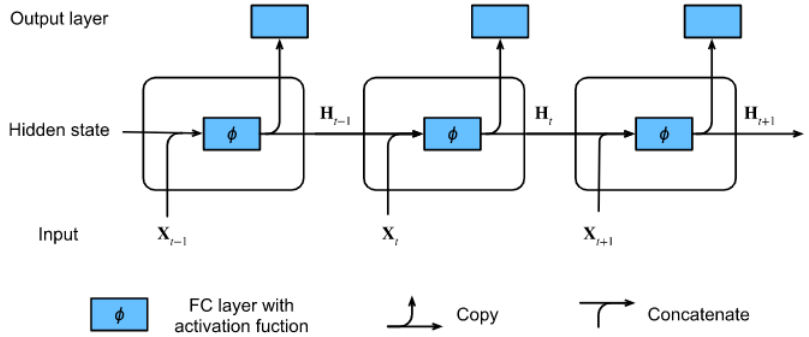

Figure 2.3: Recurrent Neural Networks [40].

2.3.2. Gated Recurrent Units

The drawback of RNNs is that RNNs have difficulty accessing relations in the long sequence. Gated Recurrent Units (GRUs) in Figure 2.4 solve the problem by introducing the gate mechanism. The gate is the vector with entries in [0,1]. The gate controls how much the neuron exploits the information from the input or the previous hidden state.

GRUs import two gates: Reset gate and Update gate. The Reset gate controls how much the network should remember the previous state. It can reset the hidden state when the model sees a discontinuous logic or a new start in the input sequence. The Update gate decides how much the current observation would update to the hidden state. It allows the networks to ignore uninformative input and maintain the long-term features in the hidden state.

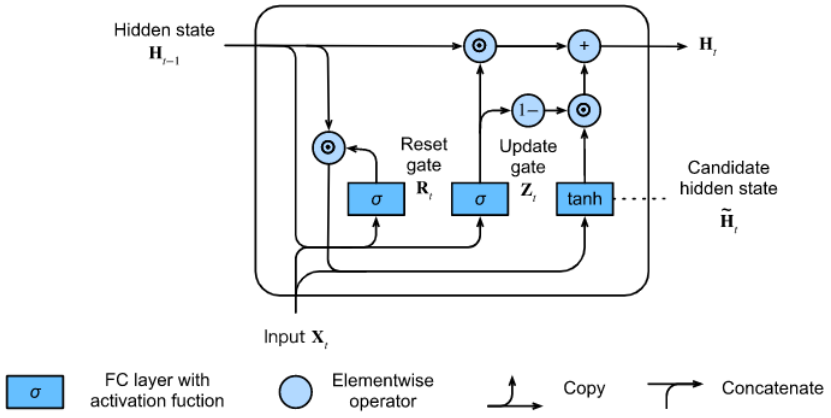

Figure 2.4: Gated Recurrent Units [40].

2.3.3. Long Short-Term Memory

Long Short-Term Memory (LSTM) [11] introduces a more complex mechanism with Forget gate, Output gate, Input gate and Memory cell in Figure 2.5. The Memory cell is another state to remember the long-term information in the top row of the network. It keeps features transporting down through the chain of the networks. LSTM exploits gates to control the input and the output of information flow from the memory cell.

First, the Forget gate is similar to the Reset gate in GRUs to reset the memory cell. Secondly, the Input gate decides how much the input information updates to the memory cell. Then, the candidate

memory proposes the features which upload to the memory cell. Last, the Output gate governs the next hidden state and the output information from the memory cell. $\odot$ in the Figure 2.5 denotes the elementwise multiplication.

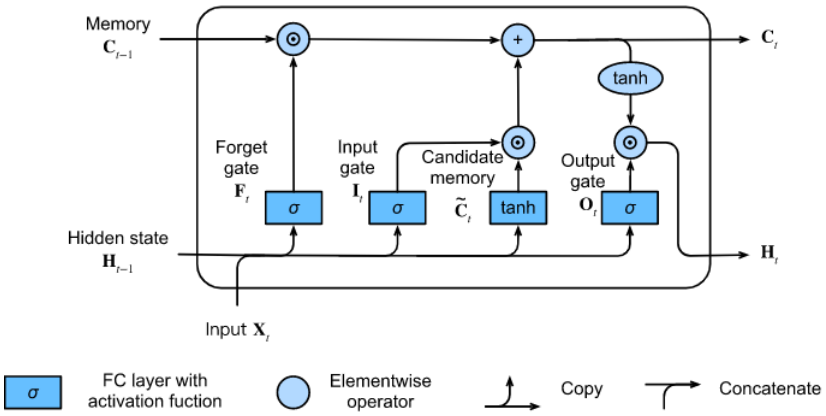

Figure 2.5: Long Short-Term Memory [40].

2.3.4. Temporal Convolutional Networks

Temporal Convolutional Networks (TCNs) stand for the model using 1D convolution to learn the sequential relation. Unlike RNNs, TCNs discard the hidden state and only use the 1D kernel with the same weights convoluting through the sequence.

TCNs take the sequence with any length and can output either at the same length or the single time step. In addition, TCNs can be causal, which means the current feature can only see information from previous time steps in Figure 2.6.

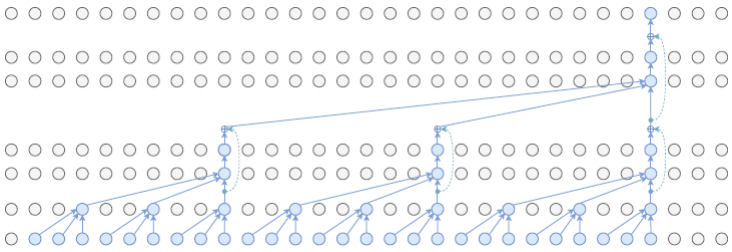

Figure 2.6: Causal convolutions [31].

TCNs explore dilated convolution to learn the long-range relationship. Dilated convolution in Figure 2.7 is a convolution operation by inserting space between elements of the kernel with a predefined dilation factor. It can increase the receptive field of a network without adding more parameters.

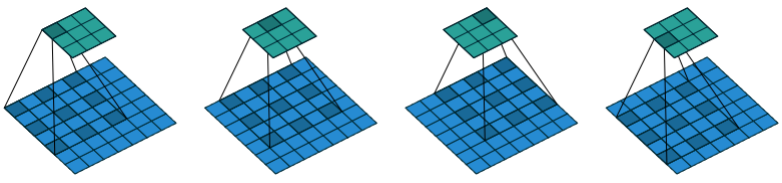

Figure 2.7: Dilated convolution with the dilation factor of 2 [8].

Compared with RNNs, where the current hidden state must wait for the update from previous steps, TCNs make the computation in parallel by applying the same kernels over the sequence. Also, the networks do not require other memory to remember states. Thus, TCNs are generally faster to train than LSTM-based RNNs [18].

2.3.5. 3D Convolutional Neural Networks

3D convolutional neural networks (3D CNNs) aim to learn spatial information and temporal features from videos with 3D kernels, which has one more dimension than 2D kernels representing time. The method takes the volume of the video as the input and convolves over both temporal and spatial dimensions in Figure 2.8.

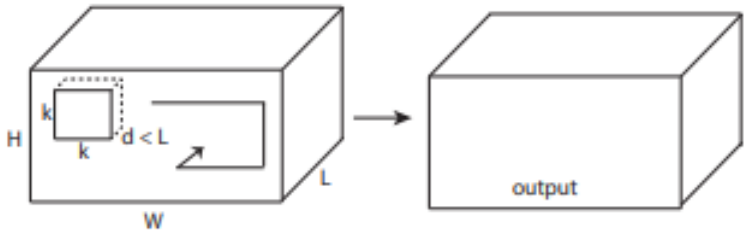

Figure 2.8: 3D Convolution with k kernel size for spatial dimension and d size for time [4]. H,W,and L stand for the height, the width and the length of the input.

3D CNNs have been proposed with different architectures, such as I3D [3], Res3D [4], C3D [37] and common architectures of ResNet [17]. For instance, I3D shows that 3D CNNs can build from 2D CNNs by repeating 2D kernels multiple times through the time dimension. This design allows the network to use the pretrained weights on ImageNet [6].

The drawback of 3D CNNs is that the networks possess more parameters than 2D CNNs and thus request a large dataset to train. Also, the method can only take a short sequence as the input because of the high computation cost.

2.3.6. Transformer

Transformer [19] utilizes self-attention mechanism to learn sequential relation. Self-attention is a mechanism to measure attention score, which is similar to human visual attention. The operation exploits three components: Query Q, Key K and Value V. The intuition is similar to the similarity measurement of the recommender method. Query is the current item to search for recommendations. Key is predefined categories of features. Each key has the corresponding value. The operation first measures the similarity between the current item and keys by the dot product. Then, the recommendation is the dot product between value and similarity.

In Transformer, self-attention first measures the similarity between the current input i and each element of the sequence by the dot product of Query and Key in Equation 2.5. The operation produces n different weights with the input sequence length n. Then, the sum of weights is normalized to 1 with the softmax in Equation 2.6. Finally, the output vector is the weighted sum of Value vectors with weights in Equation 2.7. In practice, Transformer learns Query, Key and Value with parameters W_q, W_k, W_v in Equation 2.8.

$$w'_{ij} = q_i^T k_j \tag{2.5}$$

$$w_{ij} = softmax(w'_{ij}) \tag{2.6}$$

$$y_i = \sum_j^n w_{ij} v_j \tag{2.7}$$

$$q_i = W_q x_i \quad k_i = W_k x_i \quad v_i = W_v x_i \tag{2.8}$$

The most significant difference between self-attention and models using the hidden state is that the attention mechanism treats the data as a set rather than the sequence. The attention computes features of the current stage by looking at all the input data compared with the approach of the hidden state, where the current step only sees the previous hidden state.

In practice, Transformer exploits the multi-head attention, which contains multiple attention mechanisms in Figure 2.9. The multi-head attention splits the channel sizes of input features, performs attention operation on each chunk and concatenates output back to the original dimension. For example, the input is in 512 dimensions with eight attention heads. Each attention mechanism will process

features in 64 dimensions. Then, the output features are concatenated back to 512 dimensions. The design makes the computation of the multi-head attention as fast as the single-head attention.

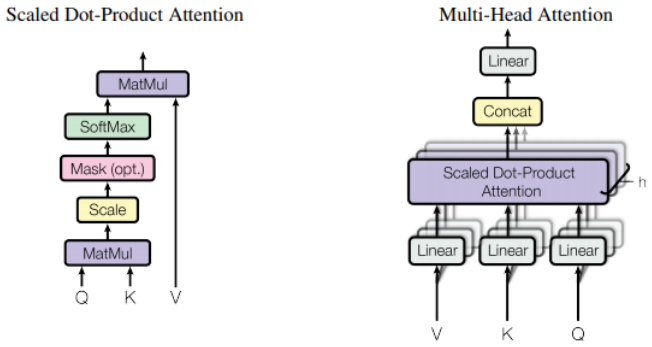

Figure 2.9: Multi-Head Attention [19].

Transformer is usually formed by multiple transformer blocks in Figure 2.10. One transforms block includes multi-head attention and feedforward networks, which is generally the fully connected network. In addition, self-attention is permutation invariant. Swapping the input sequence will generate the same prediction. A positional encoding is required to embed the position with features in the equal dimension to the input.

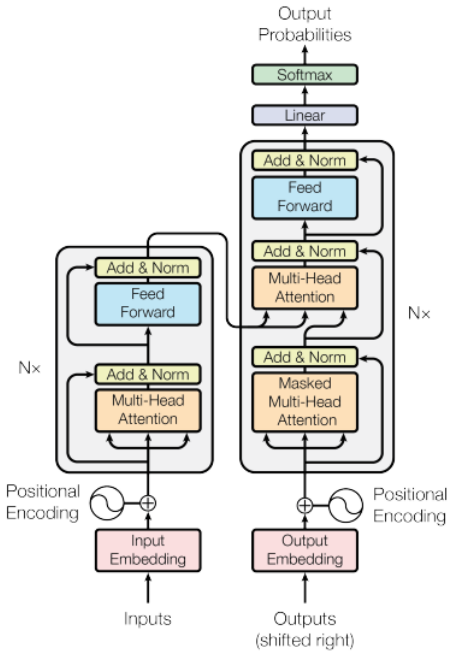

Figure 2.10: The architecture of Transformer [19].

Compared with CNNs and RNNs, Transformer has several advantages. Both CNNs and Transformer can compute in parallel. However, Transformer has less computation complexity than CNNs if the sequence length is not large. Also, Transformer has the shortest maximum path lengths than RNNs and CNNs. The maximum path length measures the maximum number of the requested walks given two points in the sequence. The shorter maximum path means that the method can see all elements of the sequence with less complexity.

3

3D Human Pose Estimation via Monocular View

3.1. Problem Description

3D human pose estimation as shown in Figure 3.1 is a task to regress 3D joint positions from video and image. The annotation of 3D joint positions is generally composed of positions from each joint center of the human body, such as the Skinned Multi-Person Linear (SMPL) annotation in Figure 3.2.

3D human pose estimation methods can further categorize into the top-down method and the button-up approach. The former detects the bounding box for each person and predicts a single person's joint positions. The latter regresses all joint positions and allocates joint positions for each person. Our research focuses on predicting the pose from a single person with a single camera. We constrain the research method to a top-down approach since we do not require methods to group joints for people. Also, the top-down approach is simpler than the bottom-up approach. As a result, the section will mainly discuss the top-down method via the monocular view.

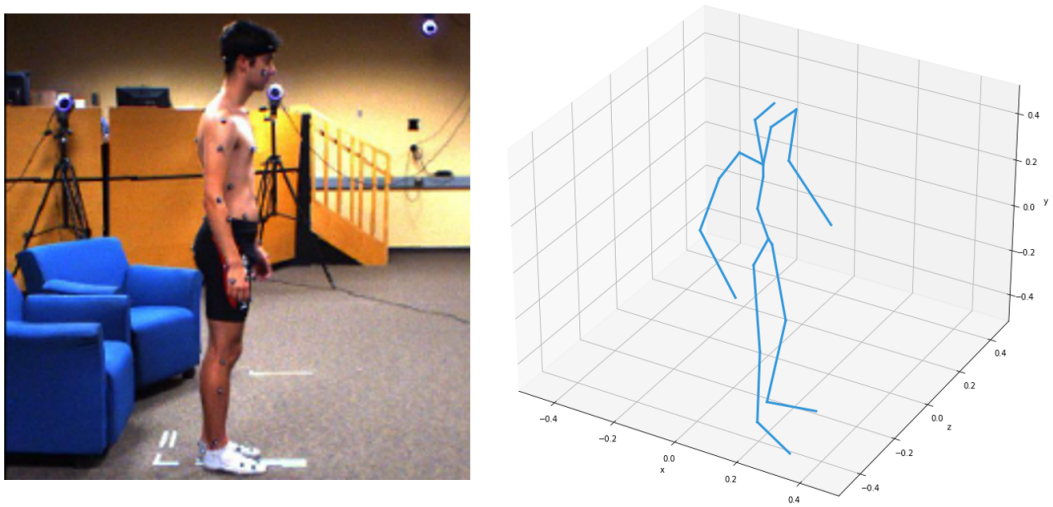

Figure 3.1: 3D human pose estimation [9]

3.2. Metrics

We introduce metrics to evaluate 3D human pose estimation methods on joint positions. The metrics for the monocular view generally evaluate the root-relative distance in the camera space. The root-relative distance is the euclidean distance after the root positions of both the prediction and the ground truth are translated into the same position. Usually, the root is the pelvis.

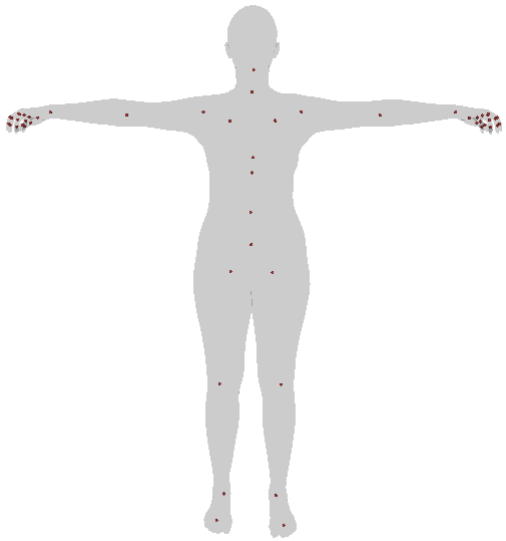

Figure 3.2: SMPL joints.

- Mean Per Joint Position Error (MPJPE):
  MPJPE is the averaged Euclidean distance between the ground truth and the prediction after aligning at the same position.
- Procrustes Analysis - Mean Per Joint Position Error (P-MPJPE):
  P-MPJPE is similar to MPJPE but applies Procrustes analysis before calculating the averaged Euclidean distance. Procrustes analysis aligns the prediction and the ground truth not only in the translation but in rotation and scaling.
- Percentage of Correct 3D Keypoints (3D PCK):
  3D PCK is the percentage of correct under a threshold. Usually, it is represented as $3D\ PCK@t\ mm$. It means that the prediction is correct if the distance between the joint position and the ground truth is within t mm.
- Mean Per Joint Velocity Error (MPJVE):
  Instead of estimating the accuracy, MPJVE measures the smoothness of the prediction through time. MPJVE is usually used when methods exploit sequence models to infer 3d human pose in the video. It is calculated by the first derivative of the 3D pose sequences.

3.3. Deep Learning Approaches

We describe current deep learning methods into three categories: CNNs-based keypoint estimation, lifting and body shape estimation.

3.3.1. CNNs-based keypoint estimation

One branch of the research builds on the pretrained CNNs to directly regress 3D joint positions from images [23, 26, 29]. We call this kind of research CNNs-based keypoint estimation.

Some methods aim to represent joint positions with 2.5D heatmaps. The 2.5D is composed of one dimension of the depth and two dimensions in the image space. The 2.5D heatmaps are usually supervised on the probability of joint position, which assumes the ground truth is a Gaussian distribution. For instance, Pavlakos et al. [30] utilize the stacked hourglass model to predict 2.5D volumetric heatmaps with the size of 64x64x64 per body joint. The drawback of 2.5D heatmap representation is that heatmaps require higher resolution to gain enough accuracy, which is costly in 3D poses [33].

Recently, soft-argmax is proposed to encode positions in lower resoluton [23, 33]. Sarandi et al. [33] combine soft-argmax and CNNs to predict joint positions in the metric scale in Fig 3.3. The heatmap

with the size of 8x8x8 per body joint is not supervised on the probability, and the whole model is trained with the root-relative loss. The benefit of the prediction in metric space rather than image space is that the model can infer the pose even if the joint is outside the image.

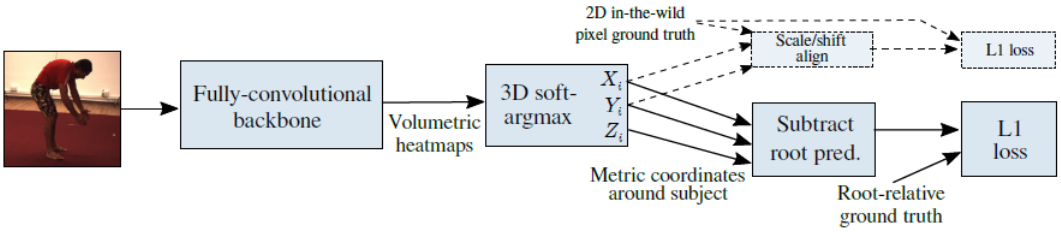

Figure 3.3: The model architecture of the metric-scale heatmap [33].

3.3.2. Lifting

Lifting is the process of lifting the 2D pose to the 3D pose. The methods belonged to this category generally exploit pretrained 2D joint detectors and design another network to lift the pose to 3D. Ideally, the whole model is still end-to-end, but 2D detectors and the lifting model are trained separately.

Typically, the lifting methods lead in the benchmark compared with the other two categories. These approaches benefit from high accurate 2D poses from pretrained 2D detectors. 2D detectors trained by the large dataset [20] have shown promising results [35]. Moreover, lifting seems a challenging task, but it is proven to be a workable concept [25].

Some methods lift the pose per image. For example, Martinez et al. [25] design a simple method with linear layer, dropout and batch normalization in Fig 3.4. Recently, research has utilized the temporal relation in the video with sequence models to smooth the prediction and solve the problem of occlusion and missing detection [4, 19, 21, 31].

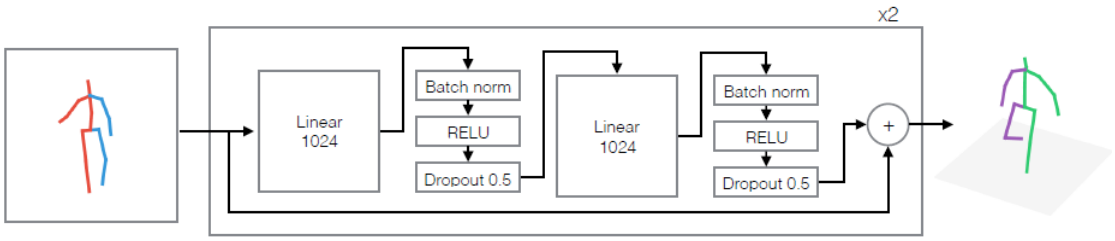

Figure 3.4: A simple yet effective baseline [25].

3.3.3. Body Shape Estimation

Some methods aim to predict not only joint position but also body shape [15, 16, 27, 38]. The approaches infer the parameters of the Skinned Multi-Person Linear (SMPL) model [22]. The SMPL is the linear model and contains two main kinds of parameters: 16 shape coefficients β and 24 joint angles θ. The output of SMPL is a triangular mesh of a human with 6890 vertices in Figure 3.5. The joint position is the weighted sum of vertices by the pretrained linear regression in the SMPL model.

These methods are supervised on either the 3D joint position or the projection of joint position on 2D images since there is no dataset for ground truth SMPL parameters. Some methods further exploit Generative Adversarial Networks (GANs) [10] to discriminate whether the prediction is natural compared to ground-truth human meshes [15, 38]. For instance, Kanazawa et al. [15] propose the Human Mesh Recovery (HMR) in Figure 3.6. The method utilizes the ResNet50 as the encoder to regress the camera parameters, body coefficients and joint angles with the supervision of the 2D projection. GANs are also used to make the prediction more natural in HMR. Although the methods of body shape estimation can infer both pose and body shape, they are more prone to pose errors than the other two kinds of methods.

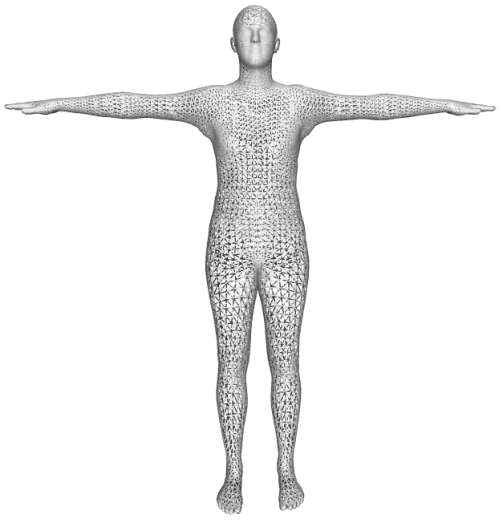

Figure 3.5: Human mesh of the SMPL model [22].

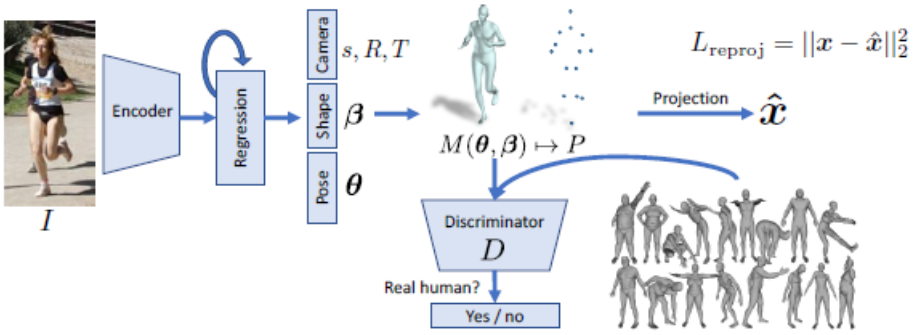

Figure 3.6: Human Mesh Recovery [15].

3.4. Datasets

This section introduces current human pose estimation datasets and SMPL datasets, such as Human 3.6M [13, 14], BMLmovi [9] and AMASS [24]. Our research exploits BMLmovi and AMASS datasets.

- Human 3.6M:
  Human 3.6M is the most widely used in 3D human pose estimation research. The data is captured with four cameras under the motion capture dataset. There are eleven subjects, and each subject performs 15 daily motions. The dataset contains around 3.6 million poses and their corresponding videos. We show example images on Human 3.6M in Figure 3.7.

- BMLmovi:
  BMLmovi dataset is a motion and video dataset collected with the inertial measurement unit, the motion capture system and four cameras. It contains 90 actors, and each subject performs 21 kinds of everyday actions. The dataset has around 9 hours of motion capture data and 17 hours of video data.

- Archive of Motion Capture as Surface Shapes (AMASS):
  AMASS unifies current motion capture datasets into 3D human meshes with Mosh++ method [22]. 3D human meshes are represented by SMPL+H parameters [32]. SMPL+H is the variant of SMPL with more parameters to represent hand motions. AMASS contains 15 different mocap datasets and around 426 hours of data with 346 subjects. We present examples of SMPL+H models on BMLmovi dataset in Figure 3.8.

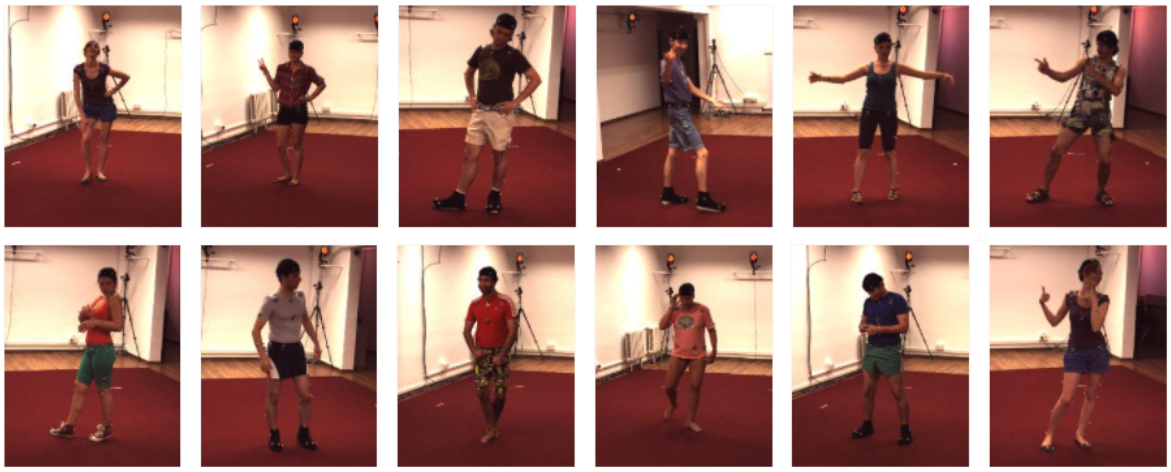

Figure 3.7: Example images of Human 3.6M. [13, 14].

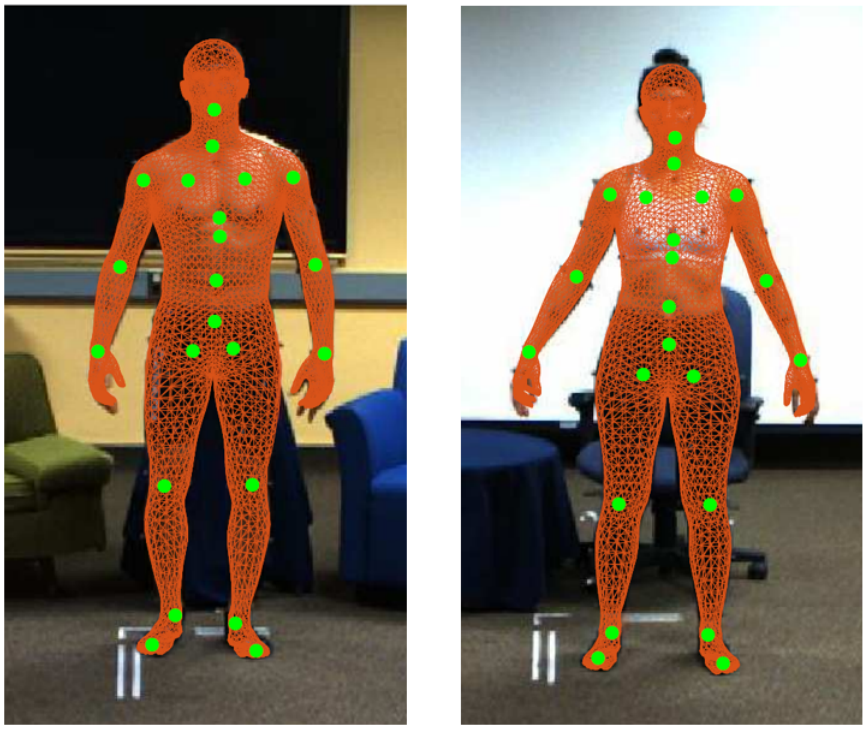

Figure 3.8: Examples of SMPL+H models on BMLmovi dataset. [9, 24].

4

OpenSim

OpenSim [5, 34] is an open-source software to help researchers develop musculoskeletal models and simulate joint angle, joint reaction and muscle forces.

4.1. Workflows

Researchers usually record motions under the motion capture system to analyze human body motion. In the system, retroreflective markers will place on the subjects' bodies, and subjects sometimes equip with the inertial measurement units on joints. Then, subjects are tracked by at least two infrared cameras during the motion.

In the process, the calibration pose (T-pose) in Fig 4.1 is recorded at the beginning. The T-pose is used for body scaling, where the default musculoskeletal model is scaled to subjects' body size. Later, subjects perform the targeting motion, and the sequence of the marker data is exploited to gain joint angel through a process called inverse kinematics. Finally, OpenSim calculates joint reaction and muscle force by other optimization methods, such as inverse dynamics and static optimization.

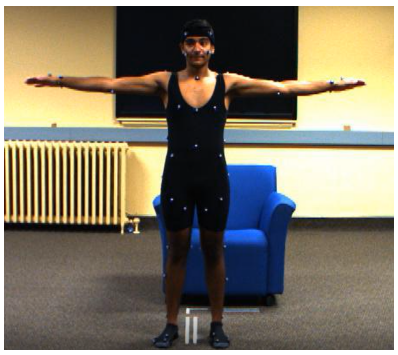

Figure 4.1: The calibration pose [9].

4.2. Musculoskeletal Model

The musculoskeletal model in Figure 4.2 is an experimental model to recreate the human bone structure and muscle connection. It can simulate muscle force and joint angle during human activities. The model in OpenSim contains five components: body, joint, kinematic constraint, marker and muscle. We focus on inferring joint angle and bone structure, so the section discusses the first four components.

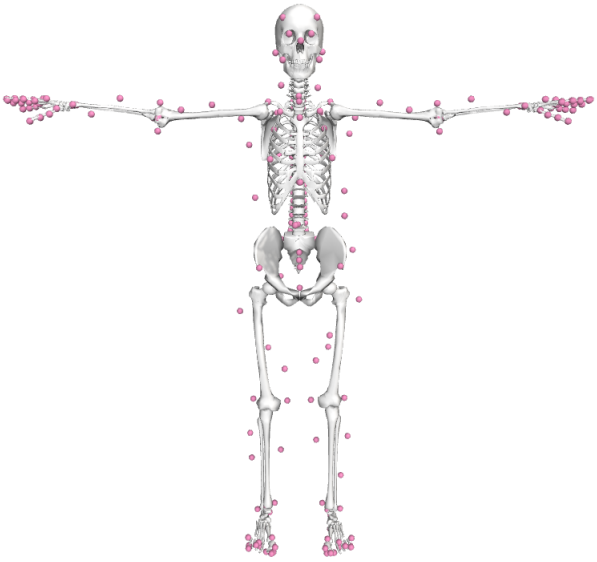

Figure 4.2: Musculoskeletal model.

4.2.1. Body

The body represents part of the human skeletal model as a rigid body and shares the same kinematic transform, such as the pelvis and the tibia. Each body has one joint and connects to its parent body. A body in OpenSim has the body scales $\beta \in R^3$. The body scales β are the ratio to the default body size on the x,y and z axes.

4.2.2. Joint

The joint manages the kinematic transform between its body and its parent body. Each joint has at most 3 degrees of freedom (DOF) for rotation along the x,y and z axes. Each DOF for rotation represents one kind of motion in the human anatomy, such as the elbow flexion. The value range of the motion is bounded in the musculoskeletal model since human motion does not possess complete freedom in 360 degrees. For instance, the value range of hip flexion is in $[-120°, 180°]$.

4.2.3. Kinematic Constraints

Kinematic constraints mean one joint angle can limit the activity of other joint angles in the human body. For example, a elbow would restrict a acromioclavicular joint. In OpenSim, the elbow is the independent joint, and the acromioclavicular joint is the dependent joint. Our research defines kinematic constraint by the Coordinate Coupler Constraint. The Coordinate Coupler Constraint creates a cubic spline function f in Equation 4.1, where the dependent joint angle y is the function output of the independent joint angle x.

$$y = f(x) \tag{4.1}$$

4.2.4. Marker

There are two types of markers: bony landmarks and tracking markers. Bony landmarks are placed on the location where bones are close to the surface and structures are touchable [7], like the elbow. Tracking markers are placed anywhere between two joint centers. They help inverse kinematics solve equations, which generally are underdetermined in the human body.

In the musculoskeletal model, the marker can be represented by a point with a vector of $\vec{d} \in R^3$ from the center of its anchor body.

4.3. Body Scaling

Body scaling is the process of scaling the default musculoskeletal model to subjects' body size. There are two types of approaches: measurement-based scaling and manual scaling. The concept of measurement-based scaling is that the subject body size is proportional to the default size with a ratio. The ratio is

estimated by comparing the distance of marker pairs between the model and the experiment.

Assume the distance $d \in R^3$ is the vector for a predefined marker pair. d_{exp} is the vector in the experimental markers. Both elements in d and d_{exp} are positive by elementwise calculating the absolute value. The ratio r is calculated by the elementwise division between d and d_{exp} in Equaton 4.3. The subject's body size $\hat{\beta}$ is proportional to the default body size β in Equation 4.2. $\oslash$ denotes the elementwise division and $\odot$ represents elementwise multiplication.

$$\hat{\beta} = r \odot \beta \tag{4.2}$$

$$r = d_{exp} \oslash d \tag{4.3}$$

Manual scaling directly assigns body scales. It is usually used when the scales are known.

The distance of bony landmarks between the model and the experiment is used to evaluate body scaling and inverse kinematics. For body scaling, the average root mean square error for the bony landmarks should be less than 1 cm [5, 34].

The drawback of body scaling is that the process is slow and manual because it is challenging to place bony landmarks on the subject body at the exact location in practice. The results of scaled musculoskeletal models would influence further applications, such as joint angle simulation and muscle force analysis. OpenSim requires researchers to manually adjust markers on the default model and check the performance of body scaling until the averaged error is less than 1 cm. As a result, the process of body scaling is iterative. Creating the scaled model becomes the slowest step in OpenSim's workflow.

4.4. Inverse Kinematics

Inverse kinematics is a process to gain joint angles without considering muscle force and physical moments [5, 34]. It obtains joint angles by minimizing the weighted least square error between markers on the model and experimental data. The practice exploits the average Euclidean distance of bony landmarks to evaluate inverse kinematics. Typically, the distance of bony landmarks should be less than two cm [5, 34].

Bibliography

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[2] Zhe Cao et al. *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. 2019. arXiv: 1812.08008 [cs.CV].

[3] Joao Carreira and Andrew Zisserman. *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset*. 2018. arXiv: 1705.07750 [cs.CV].

[4] Yu Cheng et al. *3D Human Pose Estimation using Spatio-Temporal Networks with Explicit Occlusion Training*. 2020. arXiv: 2004.11822 [cs.CV].

[5] Scott L. Delp et al. "OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement". In: *IEEE Transactions on Biomedical Engineering* 54.11 (2007), pp. 1940–1950. DOI: 10.1109/TBME.2007.901024.

[6] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[7] Farlex Medical Dictionary. *bony landmark*. URL: https://medical-dictionary.thefreedictionary.com/bony+landmark. (accessed: 13.07.2021).

[8] Vincent Dumoulin and Francesco Visin. *A guide to convolution arithmetic for deep learning*. 2018. arXiv: 1603.07285 [stat.ML].

[9] Saeed Ghorbani et al. "MoVi: A large multi-purpose human motion and video dataset". In: *PLOS ONE* 16.6 (June 2021), pp. 1–15. DOI: 10.1371/journal.pone.0253157. URL: https://doi.org/10.1371/journal.pone.0253157.

[10] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[11] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[12] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[13] Catalin Ionescu, Fuxin Li, and Cristian Sminchisescu. "Latent structured models for human pose estimation". In: *2011 International Conference on Computer Vision*. 2011, pp. 2220–2227. DOI: 10.1109/ICCV.2011.6126500.

[14] Catalin Ionescu et al. "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2014), pp. 1325–1339. DOI: 10.1109/TPAMI.2013.248.

[15] Angjoo Kanazawa et al. "End-to-end Recovery of Human Shape and Pose". In: *Computer Vision and Pattern Regognition (CVPR)*. 2018.

[16] Nikos Kolotouros et al. *Learning to Reconstruct 3D Human Pose and Shape via Model-fitting in the Loop*. 2019. arXiv: 1909.12828 [cs.CV].

[17] Okan Köpüklü et al. *Resource Efficient 3D Convolutional Neural Networks*. 2019. arXiv: 1904.02422 [cs.CV].

[18] Colin Lea et al. "Temporal Convolutional Networks for Action Segmentation and Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[19] Wenhao Li et al. *Lifting Transformer for 3D Human Pose Estimation in Video*. 2021. arXiv: 2103.14304 [cs.CV].

[20] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[21] Ruixu Liu et al. *Enhanced 3D Human Pose Estimation from Videos by using Attention-Based Neural Network with Dilated Convolutions*. 2021. arXiv: 2103.03170 [cs.CV].

[22] Matthew Loper, Naureen Mahmood, and Michael J. Black. "MoSh: Motion and Shape Capture from Sparse Markers". In: *ACM Trans. Graph.* 33.6 (Nov. 2014). ISSN: 0730-0301. DOI: 10.1145/2661229.2661273. URL: https://doi.org/10.1145/2661229.2661273.

[23] Diogo C. Luvizon, Hedi Tabia, and David Picard. *Human Pose Regression by Combining Indirect Part Detection and Contextual Information*. 2017. arXiv: 1710.02322 [cs.CV].

[24] Naureen Mahmood et al. "AMASS: Archive of Motion Capture as Surface Shapes". In: *International Conference on Computer Vision*. Oct. 2019, pp. 5442–5451.

[25] Julieta Martinez et al. *A simple yet effective baseline for 3d human pose estimation*. 2017. arXiv: 1705.03098 [cs.CV].

[26] Dushyant Mehta et al. "VNect". In: *ACM Transactions on Graphics* 36.4 (July 2017), pp. 1–14. ISSN: 1557-7368. DOI: 10.1145/3072959.3073596. URL: http://dx.doi.org/10.1145/3072959.3073596.

[27] Gyeongsik Moon and Kyoung Mu Lee. *NeuralAnnot: Neural Annotator for in-the-wild Expressive 3D Human Pose and Mesh Training Sets*. 2021. arXiv: 2011.11232 [cs.CV].

[28] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[29] Aiden Nibali et al. *3D Human Pose Estimation with 2D Marginal Heatmaps*. 2018. arXiv: 1806.01484 [cs.CV].

[30] Georgios Pavlakos et al. *Coarse-to-Fine Volumetric Prediction for Single-Image 3D Human Pose*. 2017. arXiv: 1611.07828 [cs.CV].

[31] Dario Pavllo et al. *3D human pose estimation in video with temporal convolutions and semi-supervised training*. 2019. arXiv: 1811.11742 [cs.CV].

[32] Javier Romero, Dimitrios Tzionas, and Michael J. Black. "Embodied Hands: Modeling and Capturing Hands and Bodies Together". In: *ACM Trans. Graph.* 36.6 (Nov. 2017). ISSN: 0730-0301. DOI: 10.1145/3130800.3130883. URL: https://doi.org/10.1145/3130800.3130883.

[33] Istvan Sarandi et al. "Metric-Scale Truncation-Robust Heatmaps for 3D Human Pose Estimation". In: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)* (Nov. 2020). DOI: 10.1109/fg47880.2020.00108. URL: http://dx.doi.org/10.1109/FG47880.2020.00108.

[34] Ajay Seth et al. "OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement". In: *PLOS Computational Biology* 14.7 (July 2018), pp. 1–20. DOI: 10.1371/journal.pcbi.1006223. URL: https://doi.org/10.1371/journal.pcbi.1006223.

[35] Ke Sun et al. "Deep High-Resolution Representation Learning for Human Pose Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[36] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[37] Du Tran et al. *Learning Spatiotemporal Features with 3D Convolutional Networks*. 2015. arXiv: 1412.0767 [cs.CV].

[38] Shashank Tripathi et al. *PoseNet3D: Learning Temporally Consistent 3D Human Pose via Knowledge Distillation*. 2020. arXiv: 2003.03473 [cs.CV].

[39] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[40] Aston Zhang et al. "Dive into Deep Learning". In: *arXiv preprint arXiv:2106.11342* (2021).