3D Kinematics Estimation with Biomechanics Model

Zhi-Yi Lin

Delft University of Technology

TU Delft

3D Kinematics Estimation with Biomechanics Model

by

Zhi-Yi Lin

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday July 21, 2023 at 13:00.

Student number:	5458617	
Project duration:	Sep 1, 2022 – July 21, 2023	
Thesis committee:	Dr. J. C. van Gemert,	TU Delft, Thesis advisor & Chair
	Dr. X. Zhang,	TU Delft, Daily supervisor
	Dr. P. Kellnhofer,	TU Delft, Committee member

This thesis is confidential and cannot be made public until June 1, 2024.

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

Preface

This research report documents the results of my master's thesis project "3D Kinematics Estimation with Biomechanics Model" at Delft University of Technology.

Having an interest in combining advanced computer vision techniques with healthcare applications, I consider this project as a great opportunity to explore the solutions for overcoming the challenges of data collection and to bridge the gap between the research fields of computer vision and biomechanical engineering. Hopefully, one day the integration of computer vision aids in clinical settings will become standard practice.

First, I would like to express my deepest gratitude to my daily supervisor, Dr. Xucong Zhang, for his invaluable guidance and insightful comments. His support has been instrumental in keeping me on the right track throughout this project. I also want to thank Dr. Jan van Gemert for giving me thought-provoking feedback to better wrap up the entire work and for being the chair of my committee. I would like to extend my thanks to Dr. Ajay Seth, Dr. Eline van der Kruk, Bofan Lyu, Judith Cueto Fernandez, and Akshath Ram Veeravalli Hari from the Department of Biomedical Engineering for their support in all matters related to the field of biomechanics. Special thanks to Dr. Petr Kellnhofer for his willingness to be on my thesis committee.

Finally, I would like to express my appreciation to my family and my friends. It would not have been such a great journey without your unconditional support. Thank you for being there for me in my bad and good times.

Zhi-Yi Lin
Delft, July 2023

Contents

1 Introduction

3D kinematics estimation is the study of the mechanics of human movements, with the primary aim of measuring kinematics variables such as joint angles and body segment scales. The goal is to quantify human movements for standardized and objective human motion analysis. This field has applications in robotics, entertainment, healthcare, sports sciences, and beyond. Conventional approaches rely on marker-based motion capturing. However, it is characterized by its high cost, time-consuming data acquisition, movement restrictions, marker error from skin motion, and subject discomfort caused by markers. As a result, researchers have increasingly directed their attention towards markerless motion capture methods, which only rely on visual inputs and do not require specialized expertise.

Due to the advancements in machine learning algorithms and computer hardware, deep learning has experienced a resurgence since 2010. Its capability to learn complex non-linear approximation functions from large-scale datasets has achieved breakthroughs in many computer vision tasks, including image classification, object detection, style transfer, pose estimation, action recognition, and human-computer interaction. Specifically, the achievements in 3D Human Pose Estimation (HPE) provide valuable insights into the development of markerless motion capture methods, which can be further improved through the integration of deep 3D HPE techniques and biomechanics models.

Nevertheless, some challenges remained for deep-learning-based 3D kinematics estimation with the biomechanics model. Firstly, the joint annotations in most of the datasets are anatomically wrong, leading to methods using those datasets cannot have predictions with accurate biomechanics meaning. Secondly, the scarcity of large-scale datasets due to the high cost of obtaining accurate 3D kinematics labels results in a significant performance drop when the deep-learning models are tested on in-the-wild datasets. Lastly, current 3D HPE methods tend to overlook biomechanical constraints. Therefore, the predictions often deviate from human biomechanics. This is particularly problematic for 3D kinematics estimation because it demands a higher level of biomechanical fidelity.

To overcome the challenges posed by the limited availability of large-scale datasets with accurate 3D kinematics annotations, we propose a pipeline to create synthetic data with precise biomechanics annotations by aligning the body mesh from the SMPL-X model and the biomechanics skeleton from OpenSim. The resulting dataset, named **O**penSim **D**riven **A**nimated **H**uman (ODAH), encompasses diverse variations in body shapes, clothing, lighting, and camera positions. Training deep-learning models on such a dataset are expected to yield improved performance in 3D kinematics accuracy and domain generalization. To validate this concept, we also propose an end-to-end biomechanics-aware model specifically designed for 3D kinematics estimation and train this model exclusively on ODAH.

For detailed scientific contents, readers can refer to Chapter 2. We also provide background knowledge about deep learning (Chapter 3), domain generalization (Chapter 4), 3D HPE (Chapter 5), and OpenSim (Chapter 6) for better understanding of this research work.

2
Scientific Article

3D Kinematics Estimation with Biomechanics Model

Zhi-Yi Lin
Computer Vision Lab
Delft University of Technology

Abstract

Human 3D kinematics estimation involves measuring joint angles and body segment scales to quantify and analyze the mechanics of human movements. It has applications in areas such as injury prevention, disease identification, and sports science. Conventional marker-based motion capture methods are expensive both in terms of financial investment and the expertise required. On the other hand, due to the scarcity of large-scale annotated datasets, existing markerless motion capture methods suffer from challenges including unreliable 2D keypoint detection, limited anatomic accuracy, and low generalization capability. In this work, we are the first to propose a pipeline to create synthetic data with accurate kinematics annotations by aligning the body mesh from the SMPL-X model and the biomechanics skeleton from OpenSim. The generated dataset, named ODAH, exhibits diverse variations in body shapes, clothing, lighting, and camera views. For kinematics estimation, we develop a novel biomechanics-aware model that is exclusively trained on ODAH, and directly tested on real-world data. Our extensive experiments demonstrate that the proposed approach outperforms previous state-of-the-art methods when evaluated across multiple datasets, revealing the potential for advancing the resolution of human 3D kinematics estimation.

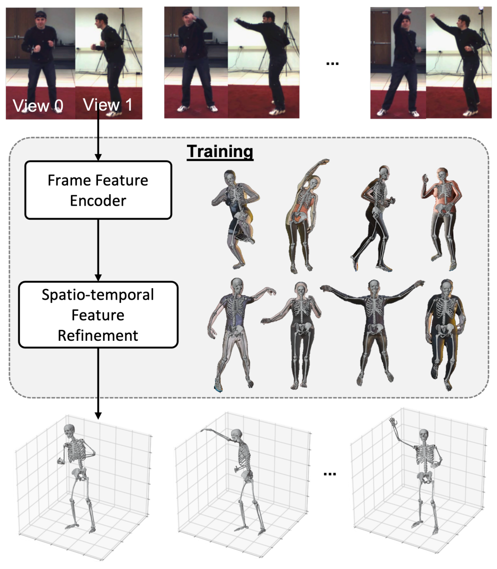


Figure 1. The proposed framework consists of a frame feature encoder and a spatio-temporal feature refinement module, which collectively infer 3D kinematics from two-view video inputs. Particularly, the end-to-end biomechanics-aware 3D kinematics estimation model is exclusively trained on synthetic data.

1. Introduction

Human 3D kinematics is the biomechanical analysis of human motion, with a primary focus on inferring kinematic variables such as joint angles and body segment scales. This field contributes to our understanding of human movements and finds applications in robotics, entertainment, healthcare, sports sciences, and beyond. The conventional approaches rely on marker-based motion capture, which has limitations such as high cost, time-consuming data acquisition, movement restrictions, marker error from skin motion, and subject discomfort caused by markers [9, 22]. To address these issues, researchers have focused on developing markerless motion capture methods that utilize visual inputs and machine learning techniques to estimate human motion without the need for physical markers and specialized expertise [1, 24, 32]. Although markerless methods offer a non-invasive and cost-effective alternative, they often follow the multi-step paradigm and heavily depend on accurate 2D body keypoint detection methods.

In recent years, deep learning has propelled substantial advancements in the field of 3D Human Pose Estimation (HPE) [5, 12, 13, 18–20, 27, 28, 36, 37, 39, 40, 42], which demonstrates significant enhancements in the performance of various computer vision tasks, including human action recognition, human activity recognition, and human-computer interaction. The achievements made in 3D HPE also provide valuable insights into the development of markerless motion capture methods, which can be further improved through the integration of deep 3D HPE techniques and biomechanics models.

However, there exist significant gaps between the two research fields. One primary concern is the prevalent use of the 2D-to-3D lifting in many 3D HPE methods [5, 18–20, 26, 27, 35, 37, 40, 42]. These methods infer 3D poses from detected 2D joints. Unfortunately, the 2D joint annotations are often anatomically wrong, resulting in imprecise 3D kinematics estimation. Another challenge is the scarcity of large-scale datasets with accurate 3D biomechanics annotations. Although some datasets incorporate markerless motion capture systems [10, 30, 32], they suffer from sensor noise and lack synchronization with the recording videos. Additionally, the expensive and time-consuming nature of marker-based motion capturing further impedes the construction of large-scale datasets necessary for deep-learning-based markerless motion capture models [1]. Last, many 3D HPE methods overlook the incorporation of biomechanical constraints, leading to predictions that deviate from human biomechanics. This discrepancy is especially problematic for 3D kinematics estimation, which demands a higher level of biomechanical fidelity.

To address the limited availability of large-scale datasets with biomechanics annotations, we propose a novel pipeline to create animated humans by aligning the SMPL-X model [25] and the OpenSim biomechanics model [29] and subsequently driving the OpenSim model using joint kinematics derived from AMASS dataset [21], which includes diverse motions captured by marker-based motion capture systems. Furthermore, this pipeline allows for unlimited augmentations in terms of body shapes, clothing, lighting, and camera positions, resulting in a large-scale synthetic dataset called **O**penSim **D**riven **A**nimated **H**uman (ODAH).

For 3D kinematics estimation, we propose a markerless motion capture framework comprising a biomechanics-aware model, as illustrated in Figure 1. The model leverages two-view videos as inputs and performs feature extraction [7], followed by a feature aggregation step for frame feature generation. Given the dynamic nature of human motion, the frame features are further refined by integrating temporal information through a transformer-based U-Net architecture. Finally, the model outputs a sequence of joint angles and a set of body segment scales.

The proposed model is trained exclusively on ODAH and is tested on ODAH and three real-world datasets. Through extensive experiments, we demonstrate that our framework outperforms three state-of-the-art markerless motion capture methods on average joint angle error across all datasets. The main contributions of this paper are:

- We are the first to propose a pipeline to create a large-scale OpenSim-annotated synthetic dataset with varied subject appearance, motions, and scene settings.
- We introduce an end-to-end biomechanics-aware 3D kinematics estimation model that predicts joint kinematics and body segment scales of an OpenSim model.
- Our experiments show that the model, exclusively trained on synthetic data, can achieve superior performance in average joint angle error across all datasets, indicating its potential for improving kinematics estimation and domain generalization.

2. Related Works

2.1. Markerless Motion Capture with OpenSim

Advancements in 3D HPE have enabled the integration of 3D HPE techniques with biomechanics models such as OpenSim [8], allowing for a comprehensive analysis of human biomechanics. Most of the existing methods use multi-step processing for kinematics estimation [24, 32]. The process starts with deriving 3D joint positions by triangulating the detected 2D landmarks [4] from multiple views. Next, the 3D joint positions are treated as marker positions in the Inverse Kinematics (IK) tool and scaling tool in OpenSim software to derive joint kinematics and body segment scales. The mapping from the detected 3D joints to the real 3D joints can be encoded in the marker offsets defined in the OpenSim model [24]. Another way is to train a model to learn the mapping function [32]. To further improve the performance, the landmark confidence scores are considered to remove low-confidence landmarks [24, 32]. Video trimming can also be applied to drop low-quality frames [32].

However, as demonstrated in [1], the intensive human intervention in the intermediate steps introduces potential error. Therefore, end-to-end solutions are preferred for robust 3D kinematics estimation. D3KE [1], an end-to-end method, utilizes CNNs to estimate joint kinematics and body segment scales for each frame from monocular videos. A lifting transformer encoder [17] is included to refine the predicted joint angles and body segment scales by incorporating temporal information.

Following the end-to-end approach, the proposed model simultaneously estimates joint kinematics and body scales based on visual inputs. Nevertheless, unlike D3KE, the backbone of the proposed network is particularly designed for HPE, two views are utilized to better handle occlusions, and the model is trained on a large-scale synthetic dataset with accurate annotations and varied augmentations.

2.2. 3D Human Pose Estimation

The two primary approaches are 2D-to-3D lifting and direct 3D estimation. The 2D-to-3D lifting approach requires feature learning from a sequence of 2D poses. This allows for the consideration of temporal information inherent in human motions. Additionally, it compensates for the loss of 3D information from monocular inputs. Temporal dilated Convolution Networks (TCNs) are widely used [5, 20, 27, 37] because they can effectively learn spatial and temporal features that are essential for lifting 2D joints and improving motion coherence. In recent years, transformers have gained popularity in handling long-range sequential data [18, 19, 40, 42]. These methods typically involve lifting the 2D pose to 3D ones through transformer-based networks, followed by spatial and temporal refinements. Multi-view 3D HPE [12, 13, 28] are also explored to overcome performance drops caused by occlusions.

The direct 3D HPE approach eliminates the reliance on 2D landmark detection by directly predicting 3D poses, offering advantages such as avoiding biases towards specific camera views and mitigating ambiguities associated with 2D landmarks [7, 36, 39, 39]. Without guidance from 2D landmarks, multi-view inputs, and temporal feature learning become necessary in such frameworks.

The proposed method follows the direct 3D approach, consisting of a frame feature encoder and a spatio-temporal refinement module. The frame feature encoder is responsible for mapping input frames into frame features, while the spatio-temporal refinement module refines these features by incorporating temporal cues. Notably, the frame feature encoder includes a stacked hourglass network pretrained using synthetic data [7] as the image feature extractor. The spatio-temporal refinement module is a transformer-based U-Net architecture, which is originally proposed for image denoise [38]. The combination of transformer and U-Net facilitates temporal feature learning across different temporal resolutions while concurrently refining local spatial features. To further ensure the model's biomechanics fidelity and prevent abnormal kinematics from extreme movements, a biomechanical constraint loss is included for supervision.

2.3. Domain Generalization

Domain Generalization (DG) emerges to address the domain shifts problems caused by the independent and identically distributed (i.i.d.) assumption in the machine learning paradigm. DG assumes that the test domain labels are inaccessible during the training process [3, 34]. DG can be achieved by data manipulation, representation learning, and optimizing learning strategy.

Data manipulation is especially important for applications that are more difficult to obtain annotations. For instance, in the context of 3D HPE, acquiring 3D annotations is expensive and time-consuming. Therefore, [21] generates animated SMPL-X model [25] correspondences to multiple MoCap datasets, offering opportunities for motion prior training and realistic motion standards as present in [15, 16, 41]. Incorporating varied textures into SMPL-X models and placing rendered subjects in diverse scenes can further enhance the visual quality of the synthetic datasets and in turn improve the performance on the target tasks [33]. SMPL-X models with 3D clothing and enhanced human-scene interactions further lead to more realistic synthetic data [2]. The authors demonstrated that pretraining models on the proposed synthetic data and subsequently fine-tuning them on real datasets yields state-of-the-art performance in 3D mesh-based human pose estimation.

Similar to 3D HPE, acquiring kinematics labels is troublesome, making domain generalization particularly important in this field. Thus, we propose a large-scale synthetic dataset with embedded biomechanics labels, and augmentations on body shapes, camera positions, lighting, and clothing for enhancing domain generalization.

3. Method

3.1. Overview

The proposed 3D kinematic estimation framework consists of two parts: a synthetic data generation pipeline and a biomechanics-aware model. To generate the animated humans with kinematics annotations, we propose a pipeline encompassing a combination of the skeleton from OpenSim [29], meshes from SMPL-X [25], textures from [33], and motions from AMASS [21]. We first bind the textured SMPL-X model with the OpenSim skeleton to create the subjects after a manual alignment of these two models. The SMPL-X is posed by the OpenSim skeleton, whose movements are driven by the joint kinematics extracted from the AMASS dataset [21]. The proposed synthetic data generation pipeline and examples are present in Figure 2.

For 3D kinematics estimation, the proposed end-to-end biomechanics-aware model first encodes each frame, captured from two different views, into a frame feature using a stacked hourglass network [23] and Multilayer Perceptron (MLP). Once the frame features of the entire sequence are obtained, a spatio-temporal feature refinement is performed to jointly consider spatial and temporal information.

3.2. Synthetic Data Creation

Body shapes and scales. To create diverse body shapes and body segment scales, subjects with varying physiques are selected from the BMLMovi subset of AMASS dataset. The body segment scales of the selected SMPL-X models are determined by manually scaling a generic OpenSim model to the SMPL-X mesh in Blender [6]. After binding the SMPL-X mesh and the scaled OpenSim model, new subjects with diverse body shapes and segment scales can be generated through the manipulation of body segment scales.
Motion. We utilize the OpenSim IK tool to obtain the ground truth joint kinematics. The required inputs to the OpenSim IK tool include marker trajectories and body segment scales. For marker trajectories, we extract the trajectories from the virtual markers that we place on the selected

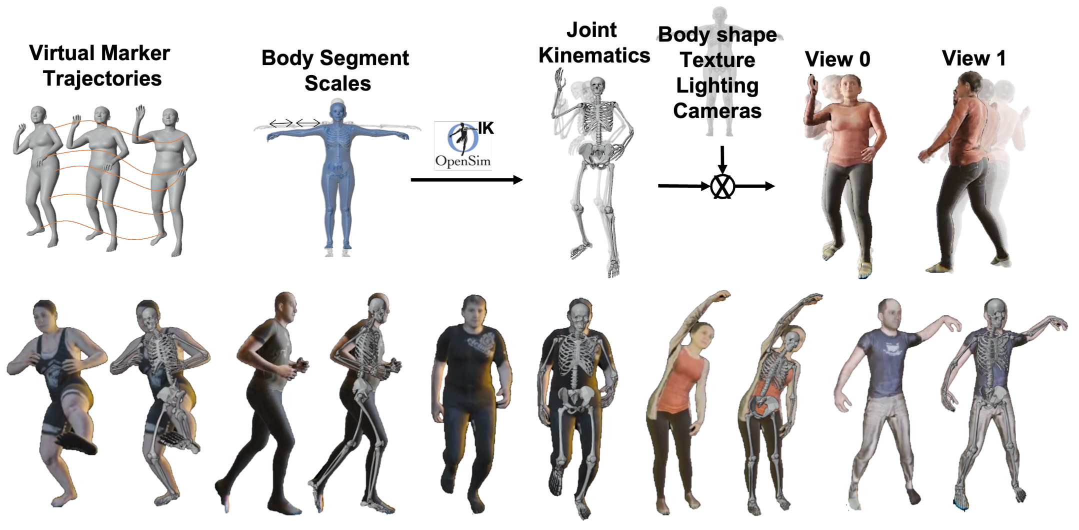


Figure 2. The proposed synthetic data generation pipeline (top) and examples with OpenSim ground truth overlays (bottom). The OpenSim IK tool requires marker trajectories and body segment scales to find the optimal joint kinematics. The marker trajectories are extracted from the AMASS dataset, and the body segment scales are measured by manually scaling the OpenSim model to the SMPL-X mesh. After combining the OpenSim skeleton with different body shapes, clothing, lighting, and camera positions, we can obtain two-view videos containing diverse animated humans.

SMPL-X model obtained from the AMASS dataset. As for the body segment scales, they are measured by manually scaling the OpenSim model to the SMPL-X mesh.

Visual quality improvements. To reduce the artifacts observed at some joints from the vanilla integration of the SMPL-X mesh and the OpenSim skeleton, we employ the pose corrective shapes defined in the SMPL-X model to make frame-by-frame adjustments to the SMPL-X mesh around the joints based on the joint rotation angles. To further preserve the volume of the SMPL-X mesh, in Blender, we introduce supplementary bones at joints with larger movements such as the shoulders, knees, and elbows.

Textures and lighting. For appearance augmentation, a variety of textured UV maps provided by [33] are utilized to generate subjects with different clothing and skin colors. The augmentation in lighting conditions includes changes in the direction of the light source and the color temperature. It is done by applying random rotations to the environment textures that we collect from open-source platforms.

Cameras. We employ two static cameras for video rendering. Both cameras are positioned at an approximate height of 1.2 $\pm$ 0.5 meters. One camera captures the frontal view, while the other one captures the sagittal view. To enhance diversity, the positions of the cameras are randomly perturbed within a small range. Both cameras are configured in landscape mode, with a fixed focal length of 26.23 mm. The sensor fit of the cameras is set to horizontal, and the sensor width is defined as 52.45 mm.

Rendering. The videos in our synthetic dataset are rendered using the Metal-accelerated BLENDER_EEVEE engine in Blender 3.5. The resolution of the videos is set to 720P, and the framerate is configured to 60 fps. Motion blur effects are enabled. The videos are encoded in the H264 format with high-quality configuration.

3.3. Network Architecture

3.3.1 Frame Feature Encoder

The biomechanics-aware model comprises a frame feature encoder and a spatio-temporal refinement module. The frame feature encoder is in charge of generating frame features from two-view frames. More specifically, the image feature of each view in a frame is extracted by a stacked hourglass network, which has demonstrated exceptional performance in HPE tasks [23]. To effectively combine the image features of two views, we first sample candidate 3D points based on the camera rays and the camera parameters. Then, N points are randomly selected from candidates whose 2D projections on both views fall within the human segmentation masks detected by YOLO [14]. Last, the point feature, denoted as $\mathbf{z}_i^{point} \in \mathbb{R}^L$ for point i, is obtained by concatenating the 3D point coordinates and the image local features allocated from two views.

To generate one compact frame feature based on the

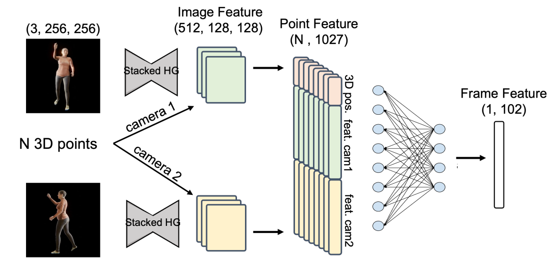


Figure 3. Architecture of the frame feature encoder. Image features are extracted by a stacked hourglass network. The locations to extract the local image features are calculated by projecting the 3D sampled point on two views. Subsequently, point features are derived by concatenating the local image features and the 3D coordinates of the sampled 3D points. Finally, MLP encodes all point features into one compact frame feature.

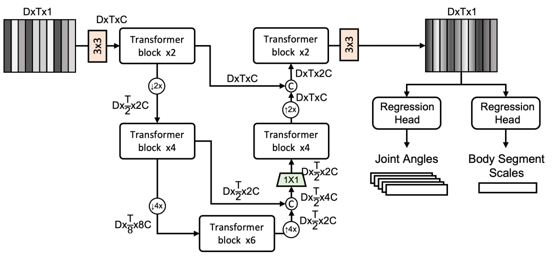


Figure 4. The transformer-based U-Net architecture for spatio-temporal feature refinement takes a sequence of frame features as inputs and outputs a sequence of joint angles and a set of body segment scales.

given N point features, we perform a two-stage feature encoding. In the first stage, each point feature is transformed into a more compact representation, denoted as $\tilde{\mathbf{z}}_i^{point} \in \mathbb{R}^{L'}$ for point i, using a shared MLP. In the second stage, the resulting N compact point features in frame j are concatenated and further encoded into a frame feature $\mathbf{Z}_j^{frame} \in \mathbb{R}^D$, for frame j, using another MLP. The architecture of the frame feature encoder is shown in Figure 3.

3.3.2 Spatio-temporal Feature Refinement

The spatio-temporal feature refinement is applied to refine a sequence of frame features with temporal information. As shown in Figure 4, this module adopts a transformer-based U-Net architecture to extract multi-range spatio-temporal features from a given sequence of frame features. The adaptations include the downsizing of the U-Net, the temporal-only downsampling and upsampling, and the removal of the skip connection from input to output.

The input to the transformer-based U-Net is generated by concatenating a sequence of frame features along the temporal axis to create a feature map, denoted as $\mathbf{Z}_0^{seq} \in \mathbb{R}^{D\times T\times 1}$, where T is the number of frames in the sequence. The feature map then undergoes processing through the transformer blocks and the U-Net architecture to extract multi-resolution spatio-temporal features along the contracting and expanding paths of the U-Net. During the contracting path, the feature map at level l, $\mathbf{Z}_l^{seq} \in \mathbb{R}^{D\times t\times c}$, is downsampled to $\mathbf{Z}_{l+1}^{seq} \in \mathbb{R}^{D\times (t/r)\times rc}$, where r represents the downsampling factor from level k to level $l+1$, t is the temporal length, and c is the number of channels. During the expanding path, refined features are generated by hierarchically combining the latent features from different levels using multiple transformer blocks. Finally, regression heads are employed to output a sequence of per-frame joint angles $\hat{\boldsymbol{\theta}} \in \mathbb{R}^C$ and the per-sequence body segment scales $\hat{\boldsymbol{s}} \in \mathbb{R}^{B\times 3}$, where C denotes the number of predicted joint angles and B represents the number of body segments.

3.4. Loss Function

For supervision, we utilize a range of factors such as joint angles, body segment scales, biomechanical constraints, and keypoints, including positions, velocities, and motion smoothness. The final loss function is written as:

$$\mathcal{L}_{total} = \mathcal{L}_{angle} + \mathcal{L}_{scale} + \mathcal{L}_{bio} + \lambda_1\mathcal{L}_{pos} + \lambda_2\mathcal{L}_{vel} + \lambda_3\mathcal{L}_{smooth} \quad (1)$$

, where λ_1 is the weight for $\mathcal{L}_{pos}$, λ_2 is the weight for $\mathcal{L}_{vel}$, and λ_3 is the weight for $\mathcal{L}_{smooth}$.

Joint angles. The joint angles refer to the coordinates defined in the generic OpenSim model. The angle loss is calculated differently based on whether the rotation of the corresponding joint is constrained.

The angle loss for free joints, denoted as $\mathcal{L}_{\boldsymbol{\theta}^f}$, is calculated as in Eq. (2). This term measures the L1 distance between the predicted and the ground truth when the angles are represented on a unit circle as commonly seen in trigonometry. This representation helps to avoid singularities and angle ambiguities caused by free rotation angles.

$$\mathcal{L}_{\boldsymbol{\theta}^f} = \frac{1}{T}\sum_{t=0}^{T-1} \parallel \hat{\boldsymbol{a}}_t - \boldsymbol{a}_t \parallel_1 \quad (2)$$

, where $\hat{\boldsymbol{a}}_t = (cos\hat{\boldsymbol{\theta}}_t^f,\ sin\hat{\boldsymbol{\theta}}_t^f)$, $\boldsymbol{a}_t = (cos\boldsymbol{\theta}_t^f,\ sin\boldsymbol{\theta}_t^f)$, $\hat{\boldsymbol{\theta}}_t^f$ and $\boldsymbol{\theta}_t^f$ are the predicted and ground truth joint angles of the free joints at time t, respectively. T is the number of frames in a sequence.

The angle loss for constrained joints, denoted as $\mathcal{L}_{\boldsymbol{\theta}^c}$, is calculated as the L1 distance between the predicted and ground truth joint angles:

$$\mathcal{L}_{\boldsymbol{\theta}^c} = \frac{1}{T}\sum_{t=0}^{T-1} \parallel \hat{\boldsymbol{\theta}}_t^c - \boldsymbol{\theta}_t^c \parallel_1 \quad (3)$$

, where $\hat{\boldsymbol{\theta}}_t^c$ and $\boldsymbol{\theta}_t^c$ are the predicted and ground truth joint angles of the constrained joints at time t, respectively. T is the number of frames in a sequence.

Biomechanical constraints. Predefined constraints are commonly used to ensure biomechanical plausibility by regulating movement. The constraints are further imposed on the network by incorporating $\mathcal{L}_{\boldsymbol{\theta}}^{bio}$ in the loss function to penalize joint angle predictions that violate the constraints as in [31]. The calculation for $\mathcal{L}_{\boldsymbol{\theta}}^{bio}$ is:

$$\begin{aligned}\mathcal{L}_{bio} = \frac{1}{T}\sum_{t=0}^{T-1} &\| \left(\hat{\boldsymbol{\theta}}_t^c \geq \boldsymbol{\theta}_{max}^c\right) \cdot \left(\hat{\boldsymbol{a}}_t - \boldsymbol{a}_{max}\right) \|_1 \\ &+ \| \left(\hat{\boldsymbol{\theta}}_t^c \leq \boldsymbol{\theta}_{min}^c\right) \cdot \left(\hat{\boldsymbol{a}}_t - \boldsymbol{a}_{min}\right) \|_1\end{aligned} \tag{4}$$

, where $\hat{\boldsymbol{a}}_t$, $\boldsymbol{a}_{min}$, and $\boldsymbol{a}_{max}$ are derived as in Eq. (2). $\hat{\boldsymbol{\theta}}_t^c$ is the predicted joint angles, and $[\boldsymbol{\theta}_{min}^c, \boldsymbol{\theta}_{max}^c]$ is the allowed range for each joint. The term T denotes the number of frames in a sequence.

Body segment scales. The body segment scale loss, denoted as $\mathcal{L}_{scale}$, is the L1 distance between predicted and ground truth body segment scales. The calculation is:

$$\mathcal{L}_{scale} = \frac{1}{B}\sum_{i=0}^{B-1} \|\hat{\boldsymbol{s}}_i - \boldsymbol{s}_i\|_1 \tag{5}$$

, where $\hat{\boldsymbol{s}}_i$ and $\boldsymbol{s}_i$ represent the predicted and ground truth body segment scales of body segment i, respectively. B is the total number of body segments.

Keypoints. We define the position of the joints and mass center of the body segments in the OpenSim model as the keypoints for loss calculation. Supervision of the position and velocity of the keypoints implicitly considers the body segment scales and the skeleton topology in 3D space. The acceleration of the keypoints is employed as a regularization term to improve motion smoothness.

Given the joint angles at frame t, denoted as $\hat{\boldsymbol{\theta}}_t \in \mathbb{R}^J$, and the body segment scales, denoted as $\hat{\boldsymbol{s}} \in \mathbb{R}^{B\times 3}$, the keypoint $\hat{\boldsymbol{P}}_t \in \mathbb{R}^{K\times 3}$ are derived as $\Phi_{forward}(\hat{\boldsymbol{\theta}}_t, \hat{\boldsymbol{s}})$, where K is the number of keypoints, $\Phi_{forward}$ is the kinematics forward function define in the OpenSim model.

The keypoint position loss $\mathcal{L}_{pos}$ is defined as the L1 distance between the prediction and the ground truth keypoint positions. The derivation is:

$$\mathcal{L}_{pos} = \frac{1}{TK}\sum_{t=0}^{T-1}\sum_{i=0}^{K-1} \|\hat{\boldsymbol{p}}_{i,t} - \boldsymbol{p}_{i,t}\|_1 \tag{6}$$

, where $\hat{\boldsymbol{p}}_{i,t}$ and $\boldsymbol{p}_{i,t}$ are the predicted and the ground truth keypoint positions, respectively. T is the number of frames, and K is the number of keypoints. Note that all the positions are relative to the pelvis.

The keypoint velocity loss $\mathcal{L}_{vel}$ is defined as the L1 distance between the velocity of the prediction and the ground truth keypoint. The calculation is formulated as:

$$\mathcal{L}_{vel} = \frac{1}{(T-1)K}\sum_{t=0}^{T-2}\sum_{i=0}^{K-1} \|\hat{\boldsymbol{v}}_{i,t} - \boldsymbol{v}_{i,t}\|_1 \tag{7}$$

, where $\hat{\boldsymbol{v}}_{i,t} = \hat{\boldsymbol{p}}_{i,t+1} - \hat{\boldsymbol{p}}_{i,t}$, and $\boldsymbol{v}_{i,t} = \boldsymbol{p}_{i,t+1} - \boldsymbol{p}_{i,t}$. T is the number of frames, and K is the number of keypoints.

The regularization term, $\mathcal{L}_{smooth}$, is the L2-norm of the keypoint acceleration:

$$\mathcal{L}_{smooth} = \frac{1}{(T-2)K}\sum_{t=0}^{T-3}\sum_{i=0}^{K-1} \|\hat{\boldsymbol{v}}_{i,t+1} - \hat{\boldsymbol{v}}_{i,t}\|_2^2 \tag{8}$$

, where $\hat{\boldsymbol{v}}_{i,t}$ and $\boldsymbol{v}_{i,t}$ are the predicted keypoint velocity, T is the number of frames, and K is the number of keypoints.

4. Experiments

4.1. Datasets

Training. For ODAH creation, we select 47 subjects from the BMLMovi [10] subset in the AMASS [21] dataset and extract 13 actions from each, including kicking, jumping jacks, stretching, running in place, jogging, crawling, walking, waving, vertical jumping, sitting down on a chair, throwing and catching, side gallop, and freestyle. Additionally, we include three actions from a small-scale lab experiment, namely sit-and-stand, sit-and-walk, and running on a treadmill. Overall, ODAH consists of 199 distinct subjects and 1583 videos. The source materials are 47 subjects and 675 trials of MoCap data. We extract 167 subjects for training, 22 subjects for validation, and 10 subjects for testing.

Testing. To evaluate the model's performance and the generalization capability in real-world settings, we conducted experiments on three real datasets, HumanEva [30], OpenCap [32], and BMLMovi [10], as well as the proposed synthetic ODAH. HumanEva includes three subjects performing actions such as boxing, walking, jogging, throwing and catching, and gesturing. It was recorded using three RGB cameras and a marker-based motion capture system. OpenCap consists of ten subjects performing actions including walking, squatting, rising from a chair, drop jumps, and the asymmetric counterparts. It was recorded using five RGB cameras and a marker-based motion capture system. OpenCap also provides processed MoCap data and OpenSim annotations. BMLMovi involves 90 subjects performing 21 actions, recorded using two cameras and a marker-based motion capture system. The selected subjects and clip names for all the testing sets are listed in Appendix A.2. Note that the 3D kinematics labels for HumanEva and BMLMovi were obtained by utilizing their corresponding SMPL-X models in AMASS dataset and OpenSim IK tool.

4.2. Implementation Details

An NVIDIA A40 GPU is employed for training and evaluation. Due to the memory limit of the GPU, we only finetune the pretrained stacked hourglass network [7] without spatio-temporal refinement. The stacked hourglass network is frozen after being integrated into the proposed network architecture. The training of the stacked hourglass network takes nine epochs, with batch size set to nine. The training of the entire network takes 8 epochs, with batch size set to 2. Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ is applied, and the learning rate is set to $1 \times 5 \times 10^{-5}$. Input frames are resized to $3 \times 256 \times 256$ before feature extraction. The loss weights are set as $\lambda_1 = 100$, $\lambda_2 = 100$, and λ_3=0.001.

We sample 500, denoted as N, 3D points for point feature extraction, and the point feature length, L, is 1027, while the reduced one, L', is 32. The final frame feature length, D, is 102. The sequence length, T, is set to 64. The U-Net consists of three encoder-decoder levels, with downsampling factors, r, set to two for level two, and four for level three. The number of transformer blocks for each level is [2, 4, 6], the number of attention heads is [1, 2, 4], and the number of channels is [48, 96, 384].

In the generic OpenSim model, the number of the joint angles J is 36. The number of body segments B is 22, and the number of joint keypoints K is 44. Our generic model has nine unconstrained joint angles, controlling the pelvis, and left and right arms. Therefore, only 17 joint angles are restricted by biomechanical constraints. The details of the generic model can be found in Appendix A.1.

4.3. Metrics

To make the evaluation focus more on the performance of the 3D joint kinematics, Procrustes alignment [11] is applied as the first step to align the global translation, rotation, and scaling between the predictions and the ground truth.

Mean Absolute Error (MAE) is used to evaluate joint angle error as

$$MAE_{angle} = \frac{1}{T}\sum_{t=1}^{T} \|\hat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t\|_1, \tag{9}$$

where $\hat{\boldsymbol{\theta}}_t$ is the predicted angles, $\boldsymbol{\theta}_t$ is the grdountruth angles, and T is the number of frames in a sequence.

Mean Per Joint Position Error with Procrustes Alignment (PA-MPJPE) is widely used in 3D HPE to measure the Euclidean distance between the predicted and the ground truth 3D joint keypoint positions. The calculation is

$$PA\text{-}MPJPE = \frac{1}{TB}\sum_{t=0}^{T-1}\sum_{i=0}^{B-1} \|\hat{\boldsymbol{p}}_{i,t}^{PA} - \boldsymbol{p}_{i,t}\|_2, \tag{10}$$

where B is the number of joint keypoints, $\hat{\boldsymbol{p}}_{i,t}^{PA}$ is the predicted joint keypoint positions relative to the root joint after Procrustes alignment, and $\boldsymbol{p}_{i,t}$ represents the ground truth joint positions relative to the root joint.

4.4. Baselines

We compare the performance of the proposed method against state-of-the-art methods of two multi-step methods Pose2Sim [24] and OpenCap [32], and one end-to-end method D3KE [1].

For Pose2Sim and OpenCap, frontal and sagittal views are taken as inputs, and the Body25 model in OpenPose [4] is the 2D landmark detection backbone. We use Pose2Sim's default Butterworth low-pass filter with a cut-off frequency of 6 Hz as the smoothing filter. For OpenCap, the cut-off frequency is set to half of the framerate. To ensure a fair comparison, we disable the landmark synchronization and the video trimming in OpenCap. For D3KE, only the frontal view is utilized as the input, and we choose the transformer-based temporal model with the sequence length set to 243 frames. Additionally, we only compare joint angles present in our generic OpenSim model and exclude arm flexion since D3KE's generic OpenSim model does not have a joint angle defined for arm flexion for the sake of fair comparison.

4.5. Comparison with State-of-the-Art

Table 1 shows the experimental results of Pose2Sim, OpenCap, D3KE, and our method on HumanEva, OpenCap, BMLMovi, and ODAH. For joint angle estimation, our method achieves an average error of 9.32 degrees across all datasets. This is equivalent to a 20% error reduction compared to the best baseline, Pose2Sim. For joint keypoint position error, although our method only shows superior results in ODAH, it still achieves the second-best average error across all datasets. Compared to Pose2Sim, the error of 61.93 mm is a 4% error increase, which is much smaller than the improvement in joint angle estimation.

More importantly, since our model is solely trained on synthetic ODAH without finetuning on any real data, the superior average results indicate the effectiveness of using synthetic data with augmentations on subject appearance and environment settings for improving domain generalization. These results also confirm the biomechanics fidelity of the proposed data generation pipeline. More detailed results can be found in Appendix A.3.

4.6. Ablation Study

4.6.1 Training Data Size

To make sure that the enhancement of domain generalization is from the dataset improvements, we train the proposed model on datasets with three different sizes, and test the resulting model on the four test datasets. The small set has 50 subjects and 333 clips, the medium set has 100 subjects and 684 clips, and the large set has 167 subjects and

	MAE_{angle} (deg.) ↓				PA-MPJPE (mm) ↓			
	Pose2Sim	OpenCap	D3KE	Ours	Pose2Sim	OpenCap	D3KE	Ours
HumanEva	12.15	10.53	15.57	**10.14**	50.66	**49.20**	87.76	62.86
OpenCap	10.02	**7.20**	12.47	10.93	69.44	**52.72**	95.64	85.78
BMLmovi	12.02	16.05	10.30	**10.24**	52.27	112.81	**49.59**	71.20
ODAH	12.82	15.55	18.43	**5.97**	65.78	84.42	168.30	**27.86**
Mean	11.75	12.33	14.19	**9.32**	**59.54**	74.79	100.32	61.93

Table 1. Comparison between Pose2Sim, OpenCap, D3KE, and our method on the joint angle error and joint keypoint error. The evaluation is performed on HumanEva, OpenCap, BMLMovi, and ODAH, respectively. The last row shows the average error across all datasets.

1306 clips. In Table 2, the average error across all datasets indicates that more data ensures better generalization for kinematics estimation. Since it is feasible to generate large synthetic data with the proposed pipeline, there is great potential for future method development.

4.6.2 Frame-based vs. Sequence-based

To investigate the effectiveness of temporal information, we perform an evaluation of frame-based and sequence-based methods. More specifically, frame-based prediction is implemented by removing the spatio-temporal refinement module in the proposed framework. From Table 3, it shows that except for the joint angle in ODAH, including temporal information improves the performance of kinematics estimation. Moreover, the improvements in the motion coherence, as indicated by Mean Per Joint Velocity Error (MPJVE), are much more significant than joint angle and joint position estimation.

4.6.3 Sequence Length

To explore the impact of the number of frames in the input sequence, we train the proposed model using sequence lengths 16, 32, and 64. Note the frame rates of the test video are 30 FPS, therefore, we evaluated the temporal windows of half second, one second, and two seconds. Table 4 shows that training with longer sequences yields better performance, and the enhancement is notably larger on motion coherence. Thus, incorporating longer temporal information is beneficial to kinematics estimation.

4.6.4 Loss Function

We further conduct incremental tests to examine the contribution of each loss term. The results are present in Table 5. Although the $\mathcal{L}_{bio}$ loss term does not lead to significant performance changes in the testing datasets, it is still included in the final loss function to mitigate abnormal kinematics that may occur during extreme movements, ensuring the robustness and reliability of the model. The implicit weights introduced by $\mathcal{L}_{pos}$ can improve the joint position error and the joint velocity error, but the joint angle error would increase. After adding $\mathcal{L}_{vel}$ and $\mathcal{L}_{smooth}$ for motion coherence improvement, the increased joint angle error is partially compensated, and the performance on joint position error and joint velocity error is further boosted. To achieve optimal performance overall metrics, we choose to include all loss terms shown here in the final loss function.

	MAE_{angle} (deg.) ↓			PA-MPJPE (mm) ↓		
	small	medium	large	small	medium	large
HumanEva	10.55	10.16	**10.14**	67.95	63.42	**62.86**
OpenCap	11.67	11.87	**10.93**	**83.61**	86.59	85.78
BMLmovi	10.48	10.41	**10.24**	71.35	**70.15**	71.20
ODAH	7.69	6.63	**5.97**	41.79	33.37	**27.86**
Mean	10.10	9.77	**9.32**	66.18	63.38	**61.93**

Table 2. Ablation study on different data sizes. The evaluation is performed on HumanEva, OpenCap, BMLMovi, and ODAH. We configure ODAH into small, medium, and large datasets.

	MAE_{angle} (deg.) ↓		PA-MPJPE (mm) ↓		PA-MPJVE (mm/s) ↓	
	Frame	Sequence	Frame	Sequence	Frame	Sequence
HumanEva	10.23	**10.14**	64.26	**62.86**	683.7	**308.0**
OpenCap	11.61	**10.93**	87.50	**85.78**	1090.3	**482.8**
BMLmovi	10.58	**10.24**	75.57	**71.20**	494.5	**254.2**
ODAH	**5.89**	5.97	28.45	**27.86**	360.5	**182.0**
Mean	9.58	**9.32**	63.95	**61.93**	657.3	**306.8**

Table 3. Ablation study on the effectiveness of temporal information. The evaluation is performed on HumanEva, OpenCap, BMLMovi, and ODAH. Frame-based prediction is the proposed network without spatio-temporal refinement, and sequence-based prediction is the same as the proposed network.

	MAE_{angle} (deg.) ↓			PA-MPJPE (mm) ↓			PA-MPJVE (mm/s) ↓		
# of frames	16	32	64	16	32	64	16	32	64
HumanEva	10.17	10.25	**10.14**	63.41	63.13	**62.86**	342.7	322.7	**308.0**
OpenCap	11.40	11.53	**10.93**	86.37	88.04	**85.78**	551.9	519.7	**482.8**
BMLmovi	10.36	10.31	**10.24**	75.36	71.84	**71.20**	294.1	270.3	**254.2**
ODAH	**5.93**	5.96	5.97	27.91	**27.33**	27.86	207.1	193.6	**182.0**
Mean	9.47	9.51	**9.32**	63.26	62.59	**61.93**	349.0	326.6	**306.8**

Table 4. Ablation study on different sequence lengths. The evaluation is performed on HumanEva, OpenCap, BMLMovi, and ODAH. We test sequence length set to 16, 32, and 64 frames.

	MAE_{angle} (deg.) ↓				PA-MPJPE (mm) ↓				PA-MPJVE (mm/s) ↓			
$\mathcal{L}_{angle}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\mathcal{L}_{scale}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\mathcal{L}_{bio}$		✓	✓	✓		✓	✓	✓		✓	✓	✓
$\mathcal{L}_{pos}$			✓	✓			✓	✓			✓	✓
$\mathcal{L}_{vel}$				✓				✓				✓
$\mathcal{L}_{smooth}$				✓				✓				✓
HumanEva	**9.80**	9.83	10.16	10.14	64.82	65.52	**62.84**	62.86	349.7	354.4	327.7	**308.0**
OpenCap	11.30	11.32	11.08	**10.93**	89.37	89.08	**85.53**	85.78	527.0	541.9	526.3	**482.8**
BMLmovi	10.28	10.26	10.32	**10.24**	77.09	76.78	74.02	**71.20**	285.6	284.3	268.2	**254.2**
ODAH	**5.68**	5.73	5.95	5.97	33.23	32.83	28.10	**27.86**	268.7	274.3	208.6	**182.1**
Mean	**9.27**	9.29	9.38	9.32	66.13	66.05	62.62	**61.93**	357.8	363.7	332.7	**306.8**

Table 5. Ablation study on the loss function. The evaluation is performed on HumanEva, OpenCap, BMLMovi, and ODAH. We incrementally test the effects of biomechanical constraints (+ $\mathcal{L}_{angle}$), the keypoint positions (+ $\mathcal{L}_{pos}$), and the keypoint velocity and motion smoothness (+ $\mathcal{L}_{vel}$ + $\mathcal{L}_{smooth}$).

5. Discussion

In this paper, we propose a pipeline to create synthetic data with biomechanics annotations and an end-to-end biomechanics-aware model that is solely trained on synthetic. The model's superior performance on joint angle estimation across all datasets demonstrates the effectiveness of using synthetic data as a large-scale database for improving kinematics estimation and enhancing domain generalization. The improved performance on kinematics estimation also validates the biomechanics fidelity of the proposed dataset generation pipeline. To conclude, this work illuminates a promising future for finetuning-free deep-learning-based markerless motion capture systems. A generalizable model is particularly beneficial for kinematics estimation in clinical settings, as it is unlikely to acquire biomechanics annotations for model finetuning.

The limitations include visual quality, variations in motions, and model size. To address these limitations, future research could focus on enhancing the visual quality of animated humans by incorporating elements such as 3D clothing, hair, and dynamic body shape, and reducing artifacts. Additionally, exploring different augmentations to determine the ones that yield the greatest performance improvement would be valuable. To improve the model's performance on unseen actions, more diverse motions can be included or simulated to enrich. Lastly, improvements in model efficiency can be achieved by finding the optimal balance between the model and dataset sizes.

References

[1] Marian Bittner, Wei-Tse Yang, Xucong Zhang, Ajay Seth, Jan van Gemert, and Frans CT van der Helm. Towards single camera human 3d-kinematics. *Sensors*, 23(1):341, 2022. 1, 2, 7

[2] Michael J. Black, Priyanka Patel, Joachim Tesch, and Jinlong Yang. BEDLAM: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2023. 3

[3] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 24, 2011. 3

[4] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019. 2, 7

[5] Tianlang Chen, Chen Fang, Xiaohui Shen, Yiheng Zhu, Zhili Chen, and Jiebo Luo. Anatomy-aware 3d human pose estimation with bone-based pose decomposition. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(1):198–209, 2021. 1, 2, 3

[6] Blender Online Community. Blender - a 3d modelling and rendering package. 3

[7] Enric Corona, Gerard Pons-Moll, Guillem Alenyà, and Francesc Moreno-Noguer. Learned vertex descent: a new direction for 3d human model fitting. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pages 146–165. Springer, 2022. 2, 3, 7

[8] 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. 2

[9] Glenn S Fleisig, Jonathan S Slowik, Derek Wassom, Yuki Yanagita, Jasper Bishop, and Alek Diffendaffer. Comparison of marker-less and marker-based motion capture for baseball pitching kinematics. *Sports Biomechanics*, pages 1–10, 2022. 1

[10] Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Niko-

laus F Troje. Movi: A large multi-purpose human motion and video dataset. *Plos one*, 16(6):e0253157, 2021. 2, 6

[11] John C Gower. Generalized procrustes analysis. *Psychometrika*, 40:33–51, 1975. 7

[12] Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I Yu. Epipolar transformers. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 7779–7788, 2020. 1, 3

[13] Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. Learnable triangulation of human pose. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7718–7727, 2019. 1, 3

[14] Glenn Jocher, Chaurasia Ayush, and Jing Qiu. Yolo by ultralytics. 4

[15] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7122–7131, 2018. 3

[16] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5253–5263, 2020. 3

[17] Wenhao Li, Hong Liu, Runwei Ding, Mengyuan Liu, and Pichao Wang. Lifting transformer for 3d human pose estimation in video. *arXiv preprint arXiv:2103.14304*, 2, 2021. 2

[18] Wenhao Li, Hong Liu, Runwei Ding, Mengyuan Liu, Pichao Wang, and Wenming Yang. Exploiting temporal contexts with strided transformer for 3d human pose estimation. *IEEE Transactions on Multimedia*, 2022. 1, 2, 3

[19] Wenhao Li, Hong Liu, Hao Tang, Pichao Wang, and Luc Van Gool. Mhformer: Multi-hypothesis transformer for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13147–13156, 2022. 1, 2, 3

[20] Ruixu Liu, Ju Shen, He Wang, Chen Chen, Sen-ching Cheung, and Vijayan Asari. Attention mechanism exploits temporal contexts: Real-time 3d human pose reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5064–5073, 2020. 1, 2, 3

[21] 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, 3, 6

[22] Lars Mündermann, Stefano Corazza, and Thomas P Andriacchi. The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications. *Journal of neuroengineering and rehabilitation*, 3(1):1–11, 2006. 1

[23] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, pages 483–499. Springer, 2016. 3, 4

[24] David Pagnon, Mathieu Domalain, and Lionel Reveret. Pose2sim: An open-source python package for multiview markerless kinematics. *Journal of Open Source Software*, 2022. 1, 2, 7

[25] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019. 2, 3

[26] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Harvesting multiple views for marker-less 3d human pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6988–6997, 2017. 2

[27] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7753–7762, 2019. 1, 2, 3

[28] Haibo Qiu, Chunyu Wang, Jingdong Wang, Naiyan Wang, and Wenjun Zeng. Cross view fusion for 3d human pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4342–4351, 2019. 1, 3

[29] Apoorva Rajagopal, Christopher L Dembia, Matthew S DeMers, Denny D Delp, Jennifer L Hicks, and Scott L Delp. Full-body musculoskeletal model for muscle-driven simulation of human gait. *IEEE transactions on biomedical engineering*, 63(10):2068–2079, 2016. 2, 3, 11

[30] Leonid Sigal, Alexandru O Balan, and Michael J Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International journal of computer vision*, 87(1-2):4, 2010. 2, 6

[31] Adrian Spurr, Umar Iqbal, Pavlo Molchanov, Otmar Hilliges, and Jan Kautz. Weakly supervised 3d hand pose estimation via biomechanical constraints. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pages 211–228. Springer, 2020. 6

[32] Scott D Uhlrich, Antoine Falisse, Łukasz Kidziński, Julie Muccini, Michael Ko, Akshay S Chaudhari, Jennifer L Hicks, and Scott L Delp. Opencap: 3d human movement dynamics from smartphone videos. *bioRxiv*, pages 2022–07, 2022. 1, 2, 6, 7

[33] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, 2017. 3, 4

[34] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022. 3

[35] Jingbo Wang, Sijie Yan, Yuanjun Xiong, and Dahua Lin. Motion guided 3d pose estimation from videos. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow,*

UK, August 23–28, 2020, Proceedings, Part XIII 16, pages 764–780. Springer, 2020. 2

[36] Size Wu, Sheng Jin, Wentao Liu, Lei Bai, Chen Qian, Dong Liu, and Wanli Ouyang. Graph-based 3d multi-person pose estimation using multi-view images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11148–11157, 2021. 1, 3

[37] Jingwei Xu, Zhenbo Yu, Bingbing Ni, Jiancheng Yang, Xiaokang Yang, and Wenjun Zhang. Deep kinematics analysis for monocular 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on computer vision and Pattern recognition*, pages 899–908, 2020. 1, 2, 3

[38] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5728–5739, 2022. 3

[39] Jianfeng Zhang, Yujun Cai, Shuicheng Yan, Jiashi Feng, et al. Direct multi-view multi-person 3d pose estimation. *Advances in Neural Information Processing Systems*, 34:13153–13164, 2021. 1, 3

[40] Jinlu Zhang, Zhigang Tu, Jianyu Yang, Yujin Chen, and Junsong Yuan. Mixste: Seq2seq mixed spatio-temporal encoder for 3d human pose estimation in video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13232–13242, 2022. 1, 2, 3

[41] Siwei Zhang, Yan Zhang, Federica Bogo, Marc Pollefeys, and Siyu Tang. Learning motion priors for 4d human body capture in 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11343–11353, 2021. 3

[42] Ce Zheng, Sijie Zhu, Matias Mendieta, Taojiannan Yang, Chen Chen, and Zhengming Ding. 3d human pose estimation with spatial and temporal transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11656–11665, 2021. 1, 2, 3

A. Supplematary Matierials

A.1. Generic OpenSim Model

The OpenSim model proposed by [29] is utilized as our generic OpenSim model. It contains 22 bodies, 22 joints, and 36 coordinates defined as joint angles to control all joint kinematics. In Figure 5, we show the joint-body pairs in our generic OpenSim model. Table 6 listed the coordinates corresponding to each joint. In our generic model, the pelvis, left arm, and right arm can freely move, resulting in nine unconstrained joint angles, while the rest 27 joint angles are restricted by biomechanical constraints.

Joint	Coordinate
ground_pelvis	pelvis_tilt, pelvis_list, pelvis_rotation , pelvis_tx, pelvis_ty, pelvis_tz
hip_r/l	hip_flextion_r/l, hip_adduction_r/l , hip_rotation_r/l
patellofemoral_r/l	knee_angle_beta_r/l
walker_knee_r/l	knee_angle_r/l
ankle_r/l	ankle_angle_r/l
subtalar_r/l	subtalar_angle_r/l
mtp_r/l	mtp_angle_r/l
back	lumbar_extension, lumbar_bending , lumbar_rotation
acromial_r/l	arm_flex_r/l, arm_add_r/l, arm_rot_r/l
elbow_r/l	elbow_flex_r/l
radioulnar_r/l	pro_sup_r/l
radiushand_r/l	wrist_flex_r/l, wrist_dev_r/l

Table 6. Joint names and their corresponding coordinate axes.

A.2. Lists of testing data

We list the testing subject and vidoe clip names in each dataset for reference. The test data list of **HumanEva** is:

- Subject 1: Box_1, Jog_1, ThrowCatch_1, Gestures_1
- Subject 2: Box_1, Jog_1, Walking_1, Gestures_1
- Subject 3: Box_1, Jog_1, ThrowCatch_1, Gestures_1

The test data list of **OpenCap** is:

- Subject 3: DJ1, DJAsym1, STS1, STSweakLegs1, Squats1, SquatsAsym1, Walking1, walkingTS2
- Subject 5: DJ2, DJAsym1, STS1, STSweakLegs1, Squats1, SquatsAsym1, Walking1, walkingTS2
- Subject 6: DJ2, DJAsym3, STS1, STSweakLegs1, Squats1, SquatsAsym1, Walking1, walkingTS2
- Subject 9: DJ1, DJAsym1, STS1, STSweakLegs1, Squats1, SquatsAsym1, Walking1, walkingTS2

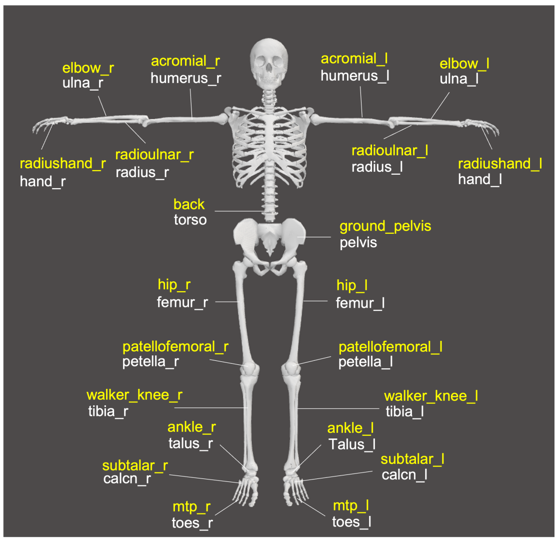


Figure 5. A generic OpenSim model and the joint-body pairs. The joints are marked in yellow. Bodies are marked in white.

- Subject 10: DJ1, DJAsym1, STS1, STSweakLegs1, Squats1, SquatsAsym1, DJ1, DJAsym5, STS1, STSweakLegs1, Squats1, SquatsAsym1, Walking2, walkingTS2

For **BMLMovi**, the test subjects are Subject 2, 13, 19, 29, 51, 72, 76, and 77. Each of them has 21 clips, namely F_1 ∼ F_21.

A.3. Breakdown of Joint Angle Error

Table 7 – Table 10 are the breakdowns of the joint angle error averaged across subjects and movements.

HumaEva	Pose2Sim		OpenCap		D3KE		Ours	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
hip_flexion	14.23	6.19	10.38	6.83	20.92	8.90	**9.23**	**5.63**
hip_adduction	4.19	1.77	4.37	1.78	4.40	**1.45**	**3.87**	1.48
hip_rotation	12.32	4.10	9.86	3.42	9.78	2.79	**7.10**	**2.39**
knee_angle	8.08	**3.05**	6.66	4.41	33.67	6.04	**8.17**	4.55
ankle_angle	12.33	3.39	**7.24**	**1.60**	9.47	3.22	7.65	2.59
subtalar_angle	7.89	2.20	8.91	2.46	9.34	2.77	**7.4**	**2.12**
mtp_angle	6.83	3.45	13.85	3.86	**4.89**	**1.55**	5.05	1.98
lumbar_extension	12.85	7.38	10.52	8.14	5.80	2.46	**5.24**	**2.22**
lumbar_bending	4.36	1.59	5.07	2.12	4.19	1.77	**3.07**	**1.38**
lumbar_rotation	6.34	6.02	8.53	**2.03**	5.29	2.49	**5.55**	2.78
arm_flex	**9.66**	**3.14**	9.74	3.64	–	–	11.95	7.37
arm_add	8.48	**1.80**	**6.61**	1.89	28.00	6.12	8.58	2.37
arm_rot	31.64	29.55	**17.46**	**5.69**	31.08	6.09	19.98	8.25
elbow_flex	12.31	8.39	**10.05**	**1.75**	16.24	8.19	13.29	5.16
pro_sup	19.33	6.65	22.34	5.87	**18.16**	**5.55**	21.53	6.3

Table 7. The per-joint joint angle error averaged across subjects and movements in HumanEva.

OpenCap	Pose2Sim		OpenCap		D3KE		Ours	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
hip_flexion	9.22	**3.76**	**8.15**	4.50	15.39	7.12	18.64	7.72
hip_adduction	7.20	2.26	**3.23**	**1.24**	4.12	1.80	4.5	1.95
hip_rotation	10.94	3.07	**4.16**	1.52	5.08	**1.42**	4.62	1.59
knee_angle	7.15	2.75	**5.73**	**1.61**	15.81	7.14	10.32	3.47
ankle_angle	16.80	27.51	**5.64**	**2.52**	8.74	3.36	7.98	2.58
subtalar_angle	6.93	2.25	**6.07**	**2.12**	8.31	2.70	8.33	3.64
mtp_angle	7.19	0.35	**0.23**	**0.26**	14.71	0.36	7.6	1.3
lumbar_extension	18.01	8.11	**9.26**	8.37	10.79	7.37	10.78	**4.49**
lumbar_bending	3.48	3.38	3.91	3.11	3.61	3.40	**3.43**	**2.52**
lumbar_rotation	**3.42**	2.06	8.09	5.27	3.76	2.19	3.69	**1.48**
arm_flex	18.34	**8.49**	**14.21**	9.16	–	–	20.52	12.47
arm_add	11.89	5.44	**9.89**	4.01	34.43	7.20	12.29	**3.51**
arm_rot	27.09	27.92	**15.07**	9.06	28.75	16.73	17.8	**7.48**
elbow_flex	**13.95**	9.22	15.72	5.87	30.86	16.49	18.38	**5.56**
pro_sup	11.87	4.14	18.58	7.46	**10.22**	**3.71**	12.06	3.99

Table 8. The per-joint joint angle error averaged across subjects and movements in OpenCap.

BMLMovi	Pose2Sim		OpenCap		D3KE		Ours	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
hip_flexion	14.98	8.97	13.99	7.89	12.00	**4.01**	**9.81**	6.94
hip_adduction	4.94	3.85	5.56	3.24	**3.08**	**1.93**	4.09	2.35
hip_rotation	11.30	3.82	8.13	3.67	**4.48**	**1.98**	7.31	4.47
knee_angle	7.19	**4.01**	**6.45**	6.15	7.43	**4.01**	8.6	4.69
ankle_angle	21.95	31.80	7.65	4.83	**6.46**	**3.18**	9.06	4.17
subtalar_angle	9.89	**3.32**	11.95	5.36	**6.69**	3.35	8.07	4.45
mtp_angle	5.45	3.99	9.68	3.61	8.09	5.18	**5.31**	**3.22**
lumbar_extension	15.49	11.29	28.10	22.83	8.29	5.25	**6.28**	**3.86**
lumbar_bending	3.36	2.58	19.09	14.28	**3.30**	**2.50**	3.59	2.6
lumbar_rotation	3.71	2.25	29.15	22.80	**3.14**	**2.21**	4.89	2.74
arm_flex	**17.81**	19.79	37.74	29.80	–	–	18.84	**18.48**
arm_add	9.30	5.96	15.90	10.41	23.18	12.49	**8.84**	**4.21**
arm_rot	25.43	16.76	35.55	22.64	36.30	17.11	**20.03**	**11.8**
elbow_flex	15.83	8.68	14.66	8.79	**7.99**	**4.13**	18.82	8.02
pro_sup	19.85	**5.82**	35.49	12.72	**18.03**	7.61	22.49	6.04

Table 9. The per-joint joint angle error averaged across subjects and movements in BMLMovi .

ODAH	Pose2Sim		OpenCap		D3KE		Ours	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
hip_flexion	19.25	11.49	25.79	21.45	26.05	10.82	**5.19**	**3.71**
hip_adduction	5.78	4.74	7.94	4.21	5.7	2.72	**2.77**	**2.08**
hip_rotation	9.37	4.17	10.3	6.43	7.62	3.06	**4.24**	**2.31**
knee_angle	12.1	9.32	14.95	10.38	40.2	17.57	**4.24**	**2.54**
ankle_angle	23.84	18.54	14.45	6.56	12.75	5.58	**5.2**	**2.6**
subtalar_angle	9.72	2.93	11.6	5.84	7.67	3.65	**4.84**	**2.7**
mtp_angle	4.11	3.89	8.58	**2.65**	8.34	4.09	**3.58**	2.9
lumbar_extension	20.52	14.4	23.81	17.52	17.4	8.4	**4.77**	**3.38**
lumbar_bending	4.76	2.83	9.85	6.8	4.95	2.99	**2.78**	**1.76**
lumbar_rotation	4.75	3.81	11.85	8.52	5.06	3.56	**3.17**	**2.4**
arm_flex	20.12	27.45	20.38	**22.04**	–	–	**11.33**	24.34
arm_add	8.74	7.18	11.67	6.54	33.94	14.65	**3.3**	**2.53**
arm_rot	23.2	17.84	25.56	18.46	32.27	19.25	**9.88**	**8.37**
elbow_flex	12.36	8.31	15.43	9.89	31.01	18.69	**6.46**	**3.97**
pro_sup	7.55	7.96	16.27	8.47	15.81	8.15	**4.65**	**4.78**

Table 10. The per-joint joint angle error averaged across subjects and movements in ODAH.

3

Deep Learning in Computer Vision

This chapter provides a brief overview of deep learning, which is a branch of machine learning that relies on artificial neural networks. Deep neural networks have more layers than traditional neural networks, allowing them to learn more complex features.

Figure 3.1 shows a simple feed-forward network. Each node is called a neuron, and the connections between them are tied with weights. The information is aggregated at each node and then passed to the next level. For example, in Figure 3.1, S_1 gets a weighted sum of x_1 and x_2, and the information at S_1 will be passed to S_3 and S_4 with different weights. The process can be rewritten as a matrix multiplication, where the matrix is composed of the weights connecting all neurons. By convention, the first layer is called the input layer, which takes the input data and passes it to the hidden layers. The hidden layers are in charge of learning latent features, which are then mapped to the target space by the output layer.

The process of learning involves adjusting the weights to minimize a loss function using gradient descent and back-propagation. Readers can refer to Chapter 4 in [44] for more details. Activation functions such as Sigmoid, ReLU, and hyperbolic tangent can introduce non-linearity into the network. Some examples are shown in Figure 3.2. Various optimizers such as SGD, RMSProp, AdaGrad, and Adam are used to improve learning efficiency and stability. A more detailed introduction can be found in Chapter 8 of [44].

There are multiple explanations for the success of deep neural networks [80, 79]. Examples are:

1. Mainfold disentangle [18]
2. Modularization [114]
3. Invariances and generalization [80]
4. Expressibility and Efficiency [64]

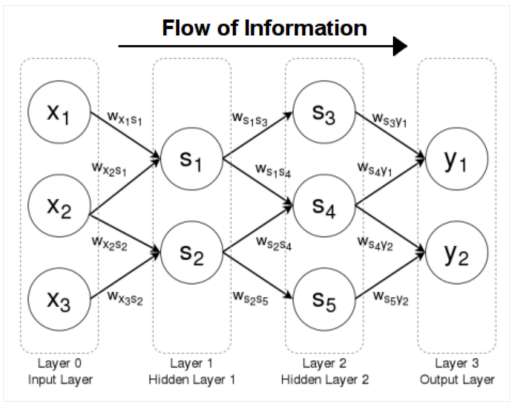


Figure 3.1: A simple feed forwrd network [39].

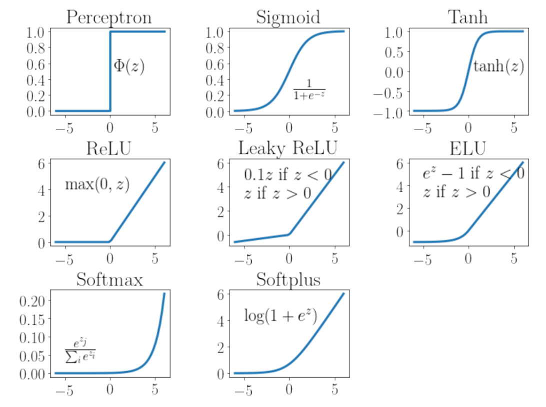


Figure 3.2: Common activation functions [50].

Most of them comprise the concept of space folding and learning hierarchical features. However, readers should keep in mind that there is no one true answer, it depends on the problem being tackled.

In the following sections, we will go through different designs of neural networks that are widely used in the computer vision research field, including the design concepts and applications.

3.1. Fully Connected Layers

Fully Connected (FC) layers, also known as Multi-Layer Perceptron (MLP), are the simplest type of feed-forward neural network. When FC layers are utilized for image feature learning, each neuron takes one pixel as input, and the output layer maps the learned features back to the target space. Figure 3.3 shows an example in handwritten digit recognition on images of size 28×28. There are 784 nodes for all pixels, and the output layer has 10 nodes representing class scores for the ten digits. The predicted class is typically the one with the highest score. Despite their simplicity, FC layers are widely used in combination with different frameworks to map learned latent features to target output space. Notably, even with such a simple design, FC layers can achieve an accuracy of almost 99% on handwritten digit classification in the MNIST dataset [35].

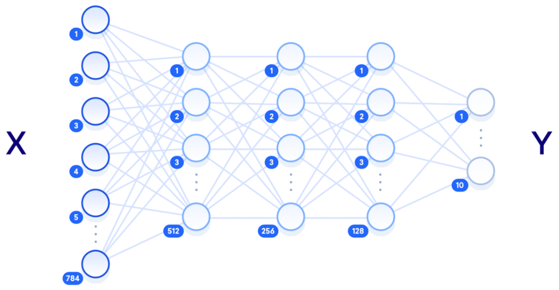


Figure 3.3: An example of handwritten digit recognition with pure FCN [1].

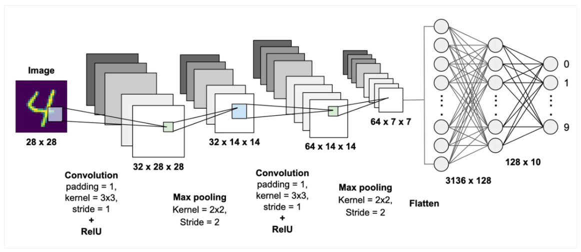


Figure 3.4: An example of handwritten digit recognition with CNN followed by FCN [2].

3.2. Convolutional Neural Networks

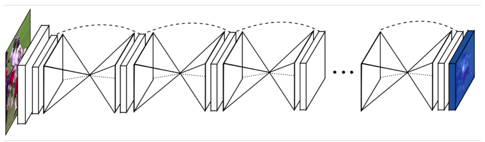
The use of FC layers for image feature learning is inefficient due to the quadratic increase in network size with respect to image width/height, and the redundancy of weights for pixels with low correlation. These inefficiencies led to the development of Convolutional Neural Networks (CNNs), which possess properties better suited for image data, such as transition invariance and localized filtering. For example, patterns in images are often repeated and have different scales, and pixels with high correlation are usually located within a local region. These similarities enable CNNs to generate good priors for image data, and the learning process becomes more efficient as kernel weights are shared across all pixels. For patterns with different scales, CNNs use pooling layers to achieve size invariance, enabling the network to learn local features at shallower layers and global features at deeper layers. A handwritten digit recognition example is shown in Figure 3.4.

Although CNNs have demonstrated state-of-the-art performance in various computer vision tasks, CNNs are not rotation invariant, meaning that the same image with different rotations is considered two different images. Additionally, since CNNs are translation invariant, it is easy to deceive the network by changing the spatial layout of an image without changing what is perceived by CNNs. This creates a loophole that provides opportunities for attackers to target existing CNN models. To address this problem, methods like adversarial learning are developed.

The following two subsections will introduce two CNN-based networks that are widely used in human pose estimation and image processing.

3.2.1. Stacked Hourglass Network

Stacked hourglass network [81] is designed for pose estimation. A single HG network (Figure 3.5b) is a convolutional encoder-decoder network capable of multi-scale feature learning, which is especially crucial for pose estimation. This is because while local evidence is essential for identifying features like faces and hands, a final pose estimate requires a coherent understanding of the full body. The person's orientation, the arrangement of their limbs, and the relationships of adjacent joints are among the many cues that are best recognized at different scales in the image. The stacked architecture (Figure 3.5a) is to enable repeated bottom-up, top-down processing used in conjunction with intermediate supervision, allowing for re-evaluation of initial estimations and features across the whole image.


(a) Multiple stacked hourglass modules.

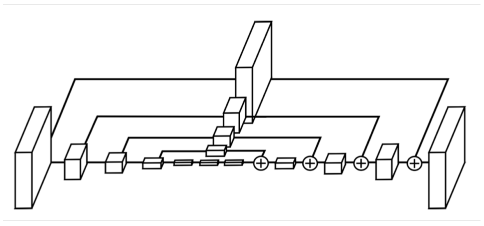

(b) A single hourglass network.

Figure 3.5: Stacked hourglass network [81].

3.2.2. U-Net

U-Net [93] is also a convolutional endoer-decoder network. The U-shaped architecture as shown in Section 3.2.2 was proposed for biomedical image segmentation. For precise localization, U-Net has a contracting path to capture context from different resolutions and a symmetric expanding path that gradually assembles features learned from different resolutions. The skip connections are crucial to preserving detailed information as features pass through the contracting and expanding path.

U-Net has achieved state-of-the-art performance in various image segmentation tasks, such as medical image segmentation, cell segmentation, and road segmentation. It has also been extended and modified to improve its performance and adapt to different tasks, for example, image restoration [23, 21, 113].

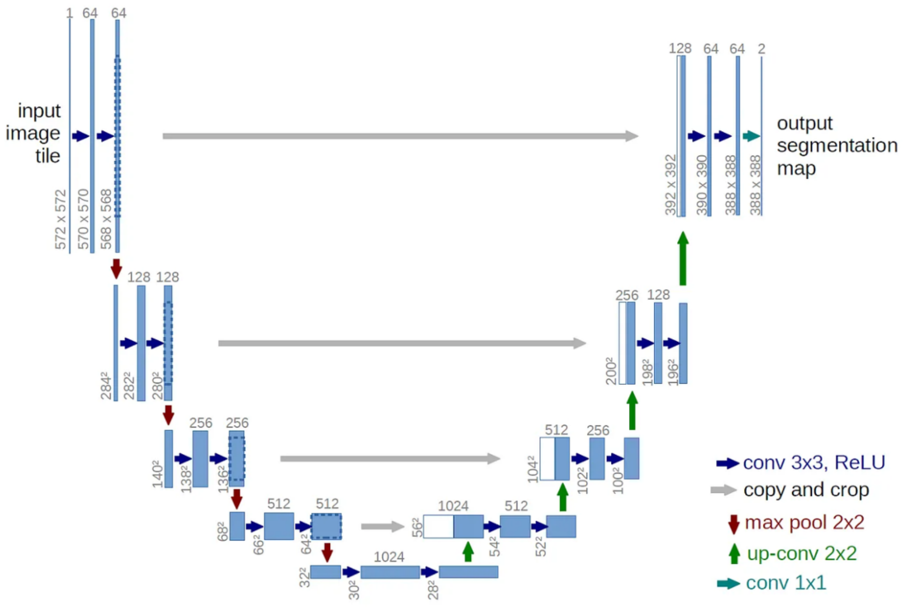


Figure 3.6: U-Net architecture [93].

3.3. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) stand for neural networks with part of the inputs from the previous outputs. A typical example of RNNs is shown in Section 3.3, where the network is unfolded over time. More specifically, the blue rectangles represent the same hidden block over time. A hidden state α is taken as the input to the hidden block at the next time step. Therefore, the network can learn features from the past, and the network size does not grow as the receptive time range increases. In computer vision, RNNs are suitable for problems that would benefit from considering information across time. For example, image captioning [73] and video summarization [118]. With proper modifications, RNNs can also be used for image classification [78].

However, RNNs are notoriously known for their difficulty in convergence due to the gradient vanish or explode problem. Without good controls over the hidden states, it is also difficult to access historical information from a long time ago. Furthermore, the computation time is longer since the data can only be processed sequentially. To overcome these issues, two more complex variations of memory units, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), were proposed.

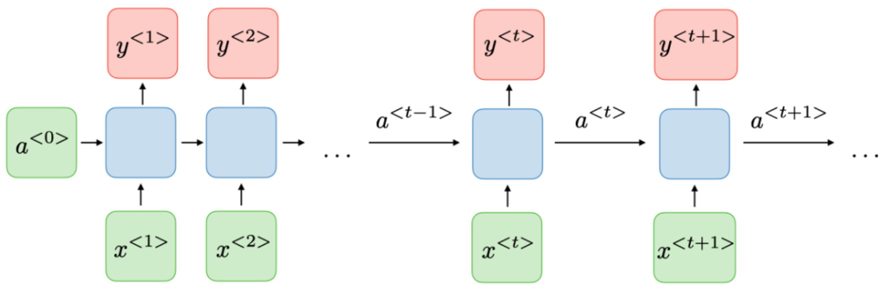


Figure 3.7: A vanilla RNNs architecture [3].

3.3.1. Long Short-Term Memory

Instead of using the hidden state as the previous information, Long short-term memory (LSTM) [47] introduces cell state ($\boldsymbol{C}_t$) as the main memory through time. In addition, a forget gate decides how much of the previous cell state will be in the current cell state. The input gate is designed to control how much of the current candidate cell state will be in the current cell state. Last, the output gate controls how much of the current cell state will be in the current hidden state. The computation is listed as:

$$\begin{aligned}
\boldsymbol{F}_t &= \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hf} + \boldsymbol{b}_f),\\
\boldsymbol{I}_t &= \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hi} + \boldsymbol{b}_i),\\
\tilde{\boldsymbol{C}}_t &= \tanh(\boldsymbol{X}_t\boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hc} + \boldsymbol{b}_c),\\
\boldsymbol{C}_t &= \boldsymbol{F}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \odot \tilde{\boldsymbol{C}}_t,\\
\boldsymbol{O}_t &= \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{ho} + \boldsymbol{b}_o),\\
\boldsymbol{H}_t &= \boldsymbol{O}_t \odot \tanh(\boldsymbol{C}_t)
\end{aligned} \tag{3.1}$$

, where $\boldsymbol{X}_t$ is the input at time t, $\boldsymbol{H}_{t-1}$ is the hidden state at the previous time step, $\boldsymbol{F}_t$ is the forget gate, $\boldsymbol{I}_t$ is the input gate, $\tilde{\boldsymbol{C}}_t$ is the candidate cell state, $\boldsymbol{C}_t$ is the cell state, $\boldsymbol{O}_t$ is the output gate, and $\boldsymbol{H}_t$ is the hidden state at the current time step. $\boldsymbol{W}_{xf}$, $\boldsymbol{W}_{hf}$, $\boldsymbol{W}_{xi}$, $\boldsymbol{W}_{hi}$, $\boldsymbol{W}_{xc}$, $\boldsymbol{W}_{hc}$, $\boldsymbol{W}_{xo}$, and $\boldsymbol{W}_{ho}$ are weight matrices and b_f, b_i, b_c, and b_o are bias vectors that are learned during training. The σ function is the sigmoid function, tanh is the hyperbolic tangent function, and $\odot$ is the element-wise multiplication operator.

To learn features from both past and future, LSTM has a bidirectional type, in which information is passed in two directions as shown in Figure 3.9. This way, a prediction can be made according to the feature learned from the past and future time steps. Examples of using LSTM for computer vision tasks are pedestrian trajectory prediction [112], traffic speed prediction [32], and so on. In general, LSTM is widely used as the baseline RNN.

3.3.2. Gated Recurrent Unit

The design of Gated recurrent unit (GRU) [27] is simpler compared to LSTM. It only has a reset gate and an update gate to control the state updates. The reset gate is used to control how much of the previous hidden state will be included in the candidate's hidden state. The update gate controls how much of the candidate's hidden state and the previous hidden state will be in the current hidden state. Compared to LSTM, GRU is easier to train and faster to run. The computation of a single GRU can be

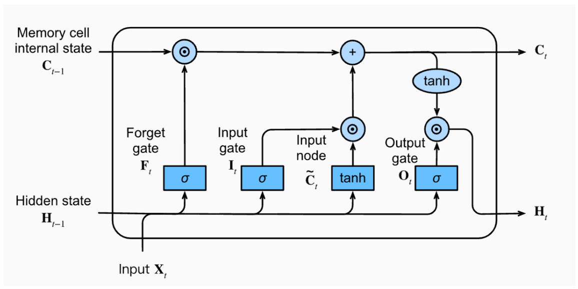


Figure 3.8: Long short-term memory cell [4].

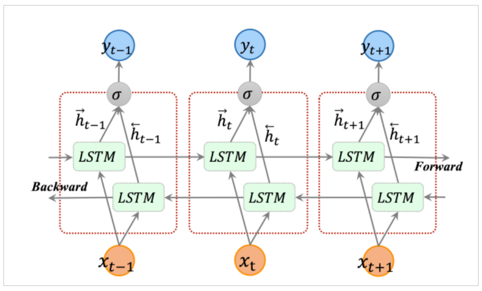


Figure 3.9: Bi-directional long short-term memory [32].

written as:

$$
\begin{aligned}
\boldsymbol{R}_t &= \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hr} + \boldsymbol{b}_r),\\
\boldsymbol{Z}_t &= \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hz} + \boldsymbol{b}_z),\\
\tilde{\boldsymbol{H}}_t &= \tanh(\boldsymbol{X}_t\boldsymbol{W}_{xh} + (\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1})\boldsymbol{W}_{hh} + \boldsymbol{b}_h),\\
\boldsymbol{H}_t &= \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1-\boldsymbol{Z}_t)\odot \tilde{\boldsymbol{H}}_t
\end{aligned}
\tag{3.2}
$$

, where $\boldsymbol{X}_t$ is the input at time step t, $\boldsymbol{H}_{t-1}$ is the hidden state at the previous time step, $\boldsymbol{R}_t$ is the reset gate, $\boldsymbol{Z}_t$ is the update gate, $\tilde{\boldsymbol{H}}_t$ is the candidate hidden state, and $\boldsymbol{H}_t$ is the updated hidden state. $\boldsymbol{W}_{xr}$, $\boldsymbol{W}_{hr}$, $\boldsymbol{W}_{xz}$, $\boldsymbol{W}_{hz}$, $\boldsymbol{W}_{xh}$, $\boldsymbol{W}_{hh}$ are weight matrices and b_r, b_z, b_h, and b_o are bias vectors that are learned during training. σ and tanh are the sigmoid and hyperbolic tangent activation functions, respectively. The $\circ$ symbol represents the element-wise multiplication.

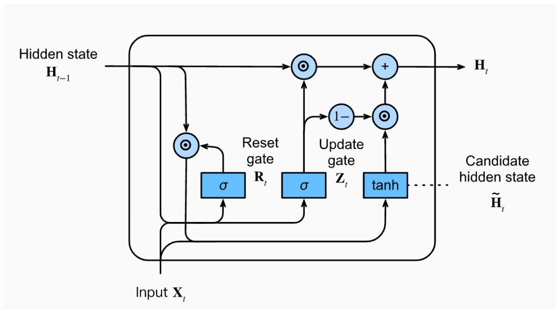


Figure 3.10: Gated recurrent unit [5].

3.4. Transformer

The transformer was first proposed in [102] for language translation tasks. The combination of self-attention layers, layer normalization, and feedforward layers overcome the RNNs' difficulty in learning long-range data dependencies.

As present in Figure 3.11, the transformer architecture consists of an encoder and a decoder. The transformer encoder can be seen as a feature learner that aggregates useful information into input tokens for the transformer decoder. It is composed of a stack of identical layers that each contain a self-attention layer and a feed-forward neural network. The self-attention layer allows the encoder to capture the relationships between different parts of the input sequence, while the feed-forward network applies a non-linear transformation to the output of the self-attention mechanism. Having a similar composition to the transformer encoder, the transformer decoder works as a decipherer, which considers the latent features from the transformer encoder and the decoder's predictions in previous time steps to make predictions.

Multi-head attention consists of h attention modules, each of which attends to different characteristics (e.g., longer-term dependencies versus shorter-term dependencies) in the input data, and all the attention heads are computed in parallel. The final output of the attention layer is a linear combination of the output from each head. The computation is written as in Equation (3.3).

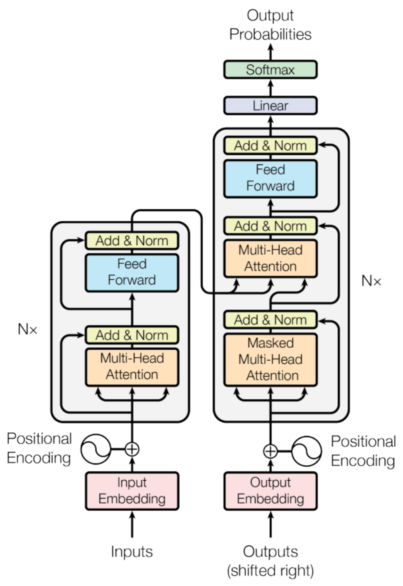


Figure 3.11: Architecture of transformer [102]. The left-hand side is the transformer encoder, and the right-hand side is the transformer decoder.

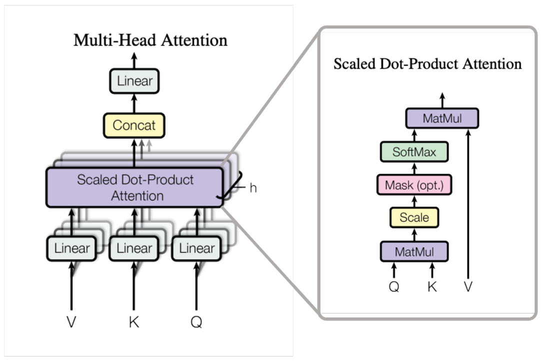


Figure 3.12: Details of multi-head attention [102]. The input sequence is first mapped to query (Q), value (V), and query (Q) for attention calculation.

$$\begin{aligned} MultiHead\left(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}\right) &= \left[head_1,\dots,head_h\right]\boldsymbol{W}_0, \\ head_i &= Attention(\boldsymbol{Q}\boldsymbol{W}_i^Q,\boldsymbol{K}\boldsymbol{W}_i^K,\boldsymbol{V}\boldsymbol{W}_i^V) \end{aligned} \tag{3.3}$$

, where $\boldsymbol{W}_0$ is a learnable matrix that combines all attention outputs, $\boldsymbol{W}_i^Q$, $\boldsymbol{W}_i^Q$, and $\boldsymbol{W}_i^Q$ are also learned to map the given query vector ($\boldsymbol{Q}$), key vector ($\boldsymbol{K}$), and value vector ($\boldsymbol{V}$) for different heads, which will be explained in the next paragraph.

Self-attention layers first maps the input sequence to three different vectors: the query vector ($\boldsymbol{Q}$), the key vector ($\boldsymbol{K}$), and the value vector ($\boldsymbol{V}$) following the computation in Equation (3.4). Figure 3.13 shows the illustration of the attention layers.

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q \qquad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K \qquad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V \tag{3.4}$$

, where the matrices $\boldsymbol{W}_K$, $\boldsymbol{W}_Q$, and $\boldsymbol{W}_V$ are learned during training.

Next, the query vector of each input will score the key vectors for each input of the sequence using the dot product. The resulting score is then treated as the weight for the value vectors, representing the information to be carried on, for each input of the sequence. Last, the output of the current input is the summation of the weighted vectors in the sequence. The whole process can be re-written to Equation (3.5)

$$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)V, \tag{3.5}$$

, where d_k is the dimension of the key vectors, and $softmax$ is the softmax function applied over the rows of the scaled dot-product of $\boldsymbol{Q}$ and $\boldsymbol{Q}^T$.

This mechanism allows the model to selectively attend to different parts of the input sequence based on their relevance to the task. Since the attention is calculated over the entire input sequence, it is particularly effective for tasks that involve long-term dependencies and non-local relationships. Moreover, the computation can be done in parallel, unlike the sequential operations in RNNs.

Despite its impressive capabilities, training transformers can be challenging and computationally demanding. Transformers typically have a large network size, requiring a substantial amount of data for

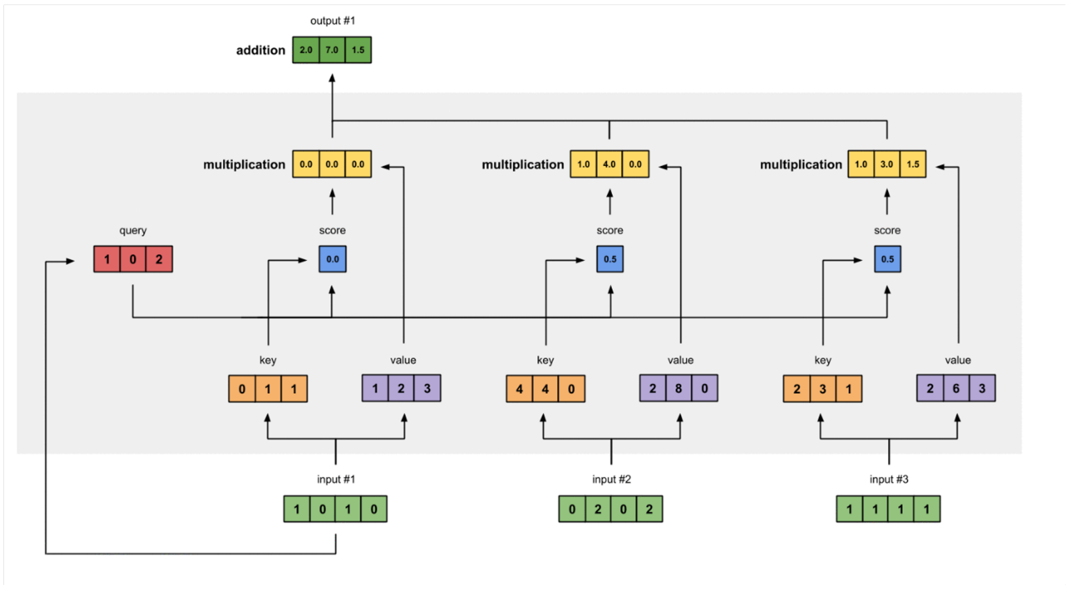


Figure 3.13: Attention layer [6]. The inputs (green) are mapped to the key (orange), value (purple), and query (red). To calculate the output of the first input, the scores derived from the query are used as weights to combine the values of all the inputs.

effective training. Without an adequate amount of dataset, transformers may not achieve comparable performance to CNN-based models, and they are also more susceptible to overfitting.

3.4.1. Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) [36] is a pre-trained neural network architecture for natural language processing (NLP) tasks. It processes input from both left to right and right to left Figure 3.14, such that the learned token can encapsulate information from both the past and future. The power of Bert-based models is that they can aggregate information from an input sequence and then handle sophisticated tasks based on the context in it. Some related works based on BERT are RoBERTa [69], ALBERT [56], and DeBERTa [46]. Commonly seen applications are question answering, sentiment analysis, named entity recognition, and natural language inference.

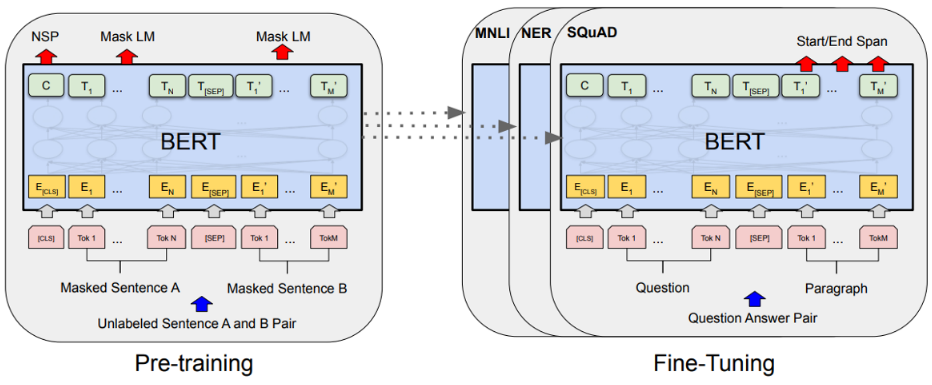


Figure 3.14: Architecture of BERT [36]. Bert takes bi-directional input to learn high-level features in the context.

3.4.2. Generative Pre-training Transformer (GPT)

The Generative Pre-trained Transformer (GPT) was proposed in [88] as a generative language model, which is a decoder-only module. Unlike most of the language models that are trained on a specific task, GPT is first trained on a large text corpus by self-supervised learning. At this stage, the model learns to predict the next word in a sentence based on the previous context. After learning from a large corpus, the model can be fine-tuned to its target tasks with a small amount of data as illustrated in Figure 3.15. Further improved models are GPT-2 [89], GPT-3 [20], and ChatGPT [7]. Commonly seen applications are language translation, text summarization, and chatbot development.

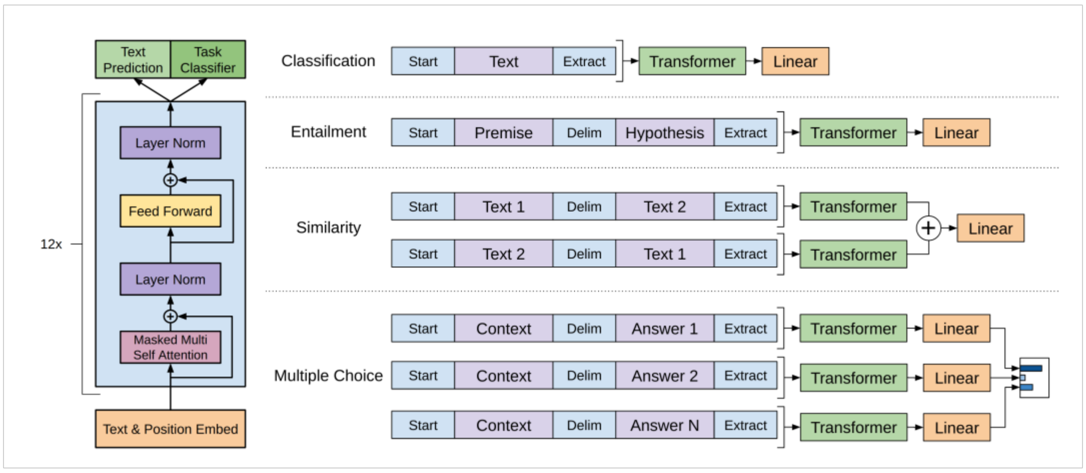


Figure 3.15: Architecture of GPT and its extended applications [88].

3.4.3. Vision Transformer (ViT)

The Vision Transformer (ViT) is a transformer-based network proposed by [37] to learn long-range image features. Compared to CNNs, ViT has a larger receptive field that allows it to learn global features at lower layers without dropping details from the input data. To apply transformers to image data, modifications such as dividing the image into patches and flattening and embedding them with their spatial positions in the original image are necessary (Figure 3.16). The resulting patch tokens are fed to a transformer encoder, resulting in the same number of encoded tokens, which are then taken as inputs of a linear layer or another neural network for downstream tasks. ViT has been successfully applied to various computer vision tasks, including video/image classification, image segmentation, and object detection [10, 70, 24, 108].

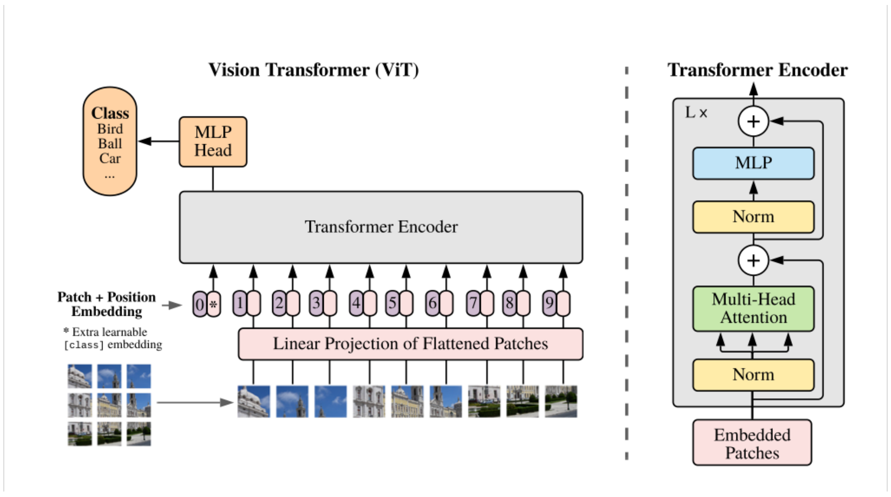


Figure 3.16: Architecture of vision transformer [37].

4 Domain Generalization

4.1. Problem Definition

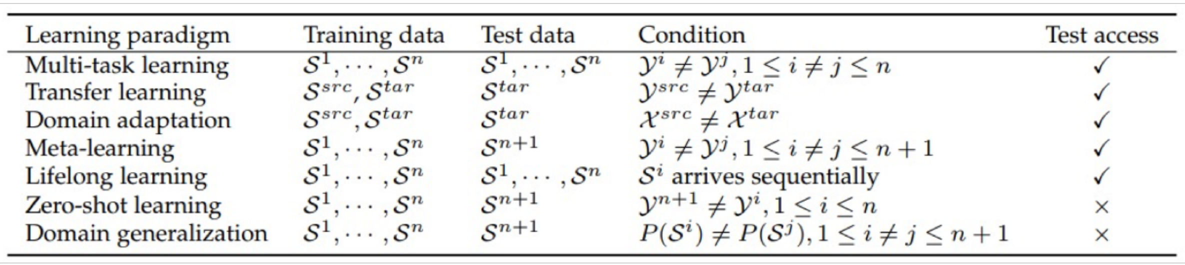
Machine learning relies on the strong assumption that data is independent and identically distributed (i.i.d.). However, in reality, this assumption often does not hold, leading to out-of-distribution (OOD) scenarios or domain shift problems. Domain shift refers to the shift in distribution between the training data (source domain) and the test data (target domain) [92]. This discrepancy can cause a significant performance drop, particularly for deep learning models due to their high capacity and flexibility in learning complex patterns.

Various approaches have been proposed to address the domain shift problem. For instance, domain adaptation has been extensively explored [14, 29, 42]. Meta-learning has also gained significant attention [40, 59, 98, 67], along with transfer learning [84, 122, 90] and other related methods. These approaches share a common objective of utilizing acquired knowledge from training data to facilitate tasks or domains with limited labeled data Figure 4.1. However, such approaches are not always feasible in numerous applications. For instance, in medical applications, it is impractical to gather prior data from each individual patient. Similarly, in biomechanics applications, the process of collecting data via marker-based motion capture systems is time-consuming and expensive.

Consequently, the concept of domain generalization (DG) emerged as a viable approach that mitigates the reliance on domain-specific information or labeled data during the training process [17]. DG methods involve training models exclusively on a collection of source domains, with the objective of capturing shared knowledge or features that can be effectively transferred across diverse domains.

4.2. Methodology

In this section, we provide an overview of the existing DG methods. The categorization shown in Figure 4.2 follows the classification proposed by [106]. Readers are encouraged to delve into further research publications for a more comprehensive understanding of DG.



Learning paradigm	Training data	Test data	Condition	Test access
Multi-task learning	$\mathcal{S}^1, \cdots, \mathcal{S}^n$	$\mathcal{S}^1, \cdots, \mathcal{S}^n$	$\mathcal{Y}^i \neq \mathcal{Y}^j, 1 \leq i \neq j \leq n$	✓
Transfer learning	$\mathcal{S}^{src}, \mathcal{S}^{tar}$	$\mathcal{S}^{tar}$	$\mathcal{Y}^{src} \neq \mathcal{Y}^{tar}$	✓
Domain adaptation	$\mathcal{S}^{src}, \mathcal{S}^{tar}$	$\mathcal{S}^{tar}$	$\mathcal{X}^{src} \neq \mathcal{X}^{tar}$	✓
Meta-learning	$\mathcal{S}^1, \cdots, \mathcal{S}^n$	$\mathcal{S}^{n+1}$	$\mathcal{Y}^i \neq \mathcal{Y}^j, 1 \leq i \neq j \leq n+1$	✓
Lifelong learning	$\mathcal{S}^1, \cdots, \mathcal{S}^n$	$\mathcal{S}^1, \cdots, \mathcal{S}^n$	$\mathcal{S}^i$ arrives sequentially	✓
Zero-shot learning	$\mathcal{S}^1, \cdots, \mathcal{S}^n$	$\mathcal{S}^{n+1}$	$\mathcal{Y}^{n+1} \neq \mathcal{Y}^i, 1 \leq i \leq n$	×
Domain generalization	$\mathcal{S}^1, \cdots, \mathcal{S}^n$	$\mathcal{S}^{n+1}$	$P(\mathcal{S}^i) \neq P(\mathcal{S}^j), 1 \leq i \neq j \leq n+1$	×

Figure 4.1: Comparison between domain generalization and the related topics [106]. S^i is the data set for domain i. Y^i is the label set for domain i. $P(S^i)$ is the data distribution of data set in domain i.

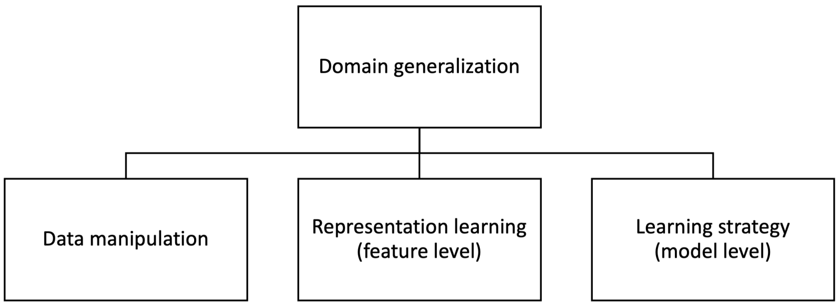

Domain generalization
Data manipulation
Representation learning (feature level)
Learning strategy (model level)

Figure 4.2: The general categorization of domain generalization methods, adopted from [106].

4.2.1. Data Manipulation

Data manipulation can be further classified into two branches. The first branch is data augmentation, also known as image transformation. Data augmentation is a common practice used during model training for many vision tasks. It involves introducing random variations to the input data [95, 55] by gamma adjustments for color images, modifications in lighting, noise addition, rotation, flipping, and more. Some examples are shown in Figure 4.3. When the target domain is known, specific augmentations can be applied. For instance, in medical image processing, noise or color bias introduced by scanners from different medical centers can be simulated. It is important to note that data augmentation should be executed with caution, as it may lead to confusing labels. For example, in the case of hand digit recognition, flipping an image may yield digits that do not exist. Rather than using a predefined policy for data augmentation, some works leverage adversarial learning [9, 31, 30] to learn augmentation policy for better generalization capabilities.

The second branch is data generation, which focuses on generating a substantial and diverse dataset to improve the generalization of models on unseen data [101, 16]. Most of the effort put into this branch is to improve the visual quality of the synthetic data (Figure 4.4). Adversarial augmentation techniques [9, 103, 105, 58] are widely used to incorporate a discriminator that in turn improves the authenticity of the synthetic data.

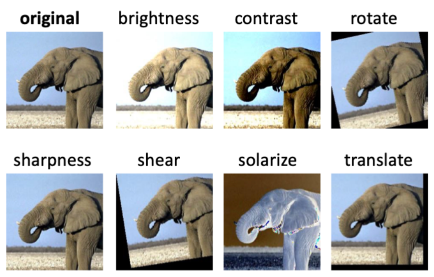


Figure 4.3: Examples of data augmentation [121].

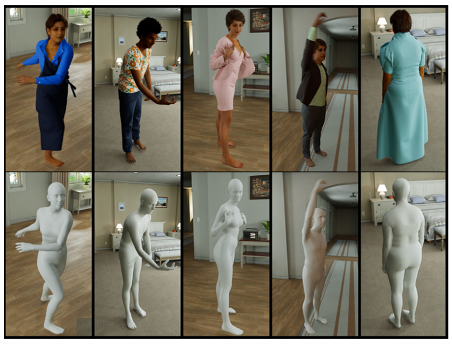


Figure 4.4: Synthetic data for human pose estimation [16].

4.2.2. Representation Learning

The second category consists of two branches: domain invariant feature learning and feature disentanglement. Domain invariant feature learning aims to enhance the similarity of features across different domains [77, 26, 104, 57]. A common approach is to incorporate the similarity between features extracted from different domain sources into the final loss function. Contrastive learning is often applied to prevent feature collapse in limited domain source [77, 26]. This technique further includes an inductive bias to move the feature embeddings apart from each other across different samples within the same domain, as depicted in Figure 4.5.

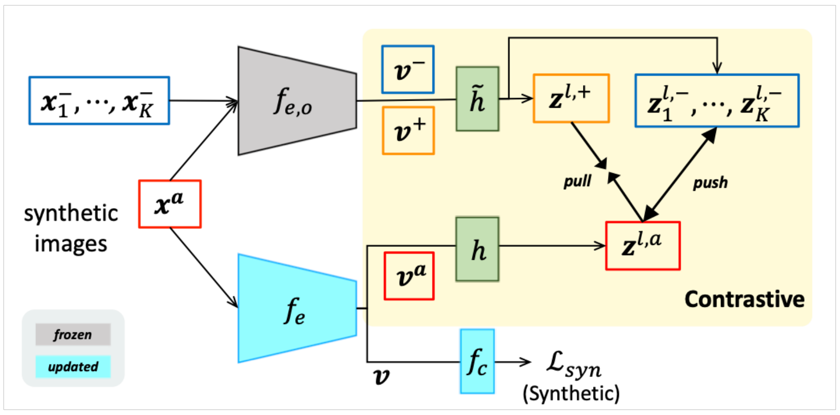


Figure 4.5: The framework proposed by [26]. Domain invariant features are learned by simultaneously **pulling** together the features embeddings from models trained on the synthetic data (blue) and the model trained on real datasets (gray), while also **pushing** away feature embeddings for different images in the same domain.

Regarding feature disentanglement, the objective is to separate components that are shared across domains and domain-specific. In [94], the model learns domain-shared and domain-specific feature embeddings and then uses the combination of these two feature sets to trick the network into perceiving nonexistent data as part of the training set. [119] disentangles the features of individual frames into groups of sub-features, each corresponding to specific semantic attributes (e.g., head, bag, shoes, etc.). These sub-features are subsequently weighted based on attribute recognition confidence and are aggregated across the temporal domain to form the final representation.

Another direction is to utilize domain-shared and domain-specific features for data generation [54, 60, 115]. For instance, [115] proposes a generator that employs both domain-shared and domain-specific features for data generation. The domain-shared features are learned using a variation encoder, while the domain-specific features are learned using a semantic encoder. Figure 4.6 provides an illustration of the data generated by combining domain-shared and domain-specific features.

4.2.3. Learning Strategy

Learning Strategy aims to adjust the training process to improve the generalization capability of the model. One well-known method is model ensembling, which involves combining multiple weak learners to create a stronger one. Examples are Random Forest [19], Adaboost [41], and voting ensembling [11]. In the context of deep learning, the same principle applies. For instance, predictions from models trained on different domains can be combined using weighted averaging, as demonstrated in [33].

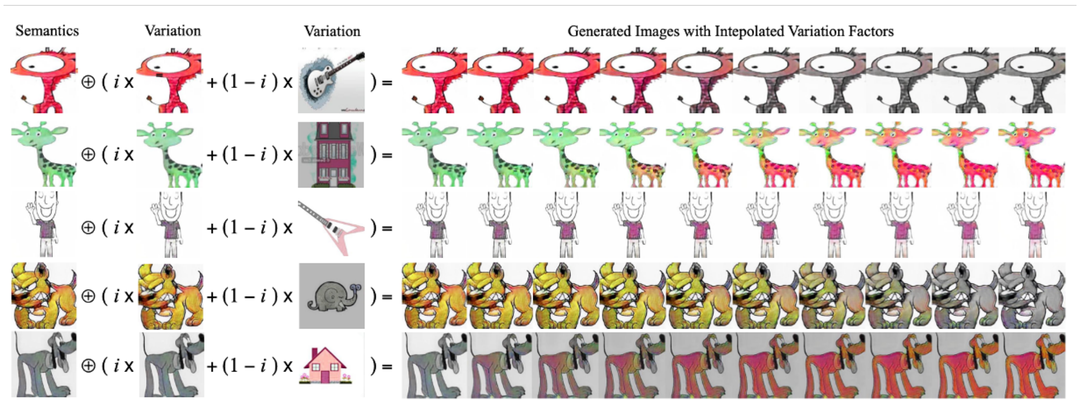


Figure 4.6: The augmented data generated by combining domain-specific and domain-shared features as proposed in [115].

5 Deep 3D Human Pose Estimation

The goal of 3D Human Pose Estimation (HPE) in the computer vision research field is to estimate 3D poses based on visual inputs. The underlying 3D poses can be poses of a shaped-based human model or a skeleton-based human model. In recent years, there has been a notable increase in attention toward 3D HPE following milestones achieved in 2D HPE. This shift can be attributed to the advantages offered by 3D HPE, such as its ability to capture more realistic human poses, and the widespread applications, including human-computer interaction, human activity recognition, autonomous driving, healthcare, and sports performance analysis.

Compared to 2D HPE, 3D HPE has challenges including depth ambiguity, perspective bias, and the scarcity of datasets with 3D annotations. Depth ambiguity refers to situations where different 3D poses can result in the same 2D pose when projected onto 2D images. Mitigating this challenge often requires additional inputs (e.g., multi-view images) or temporal information. Another approach involves considering pose priors.

The limited availability of datasets with 3D annotations poses another obstacle in 3D HPE. This scarcity restricts the potential of deep learning techniques. Consequently, instead of directly estimating 3D poses, most existing 3D HPE methods rely on lifting 2D poses to 3D poses. Moreover, the existing datasets with 2D annotations usually deviate from real human anatomy, leading to potential limitations when applying learned 3D human kinematics in healthcare applications.

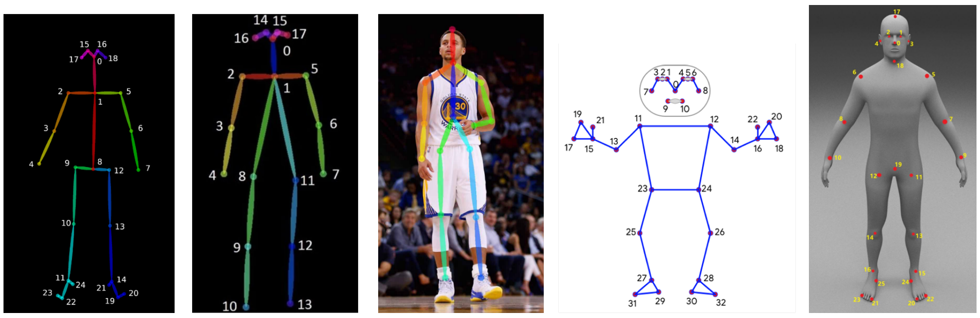
This chapter provides an overview of the current research advancements in 3D HPE and how researchers are tackling the aforementioned challenges. The chapter begins by defining branches of 3D HPE in Section 5.1. Subsequently, existing methods are introduced. Section 5.2 delves into research works specifically focusing on 3D HPE with musculoskeletal models. To assess the performance of 3D HPE methods, Commonly used evaluation metrics for 3D HPE are described in Section 5.3. Section 5.4 provides an overview of popular datasets in 3D HPE research.

5.1. 3D HPE Overview

We categorize all 3D HPE methods based on three indices: 1) the types of human modeling, 2) 3D prediction methodology, and 3) input data types.

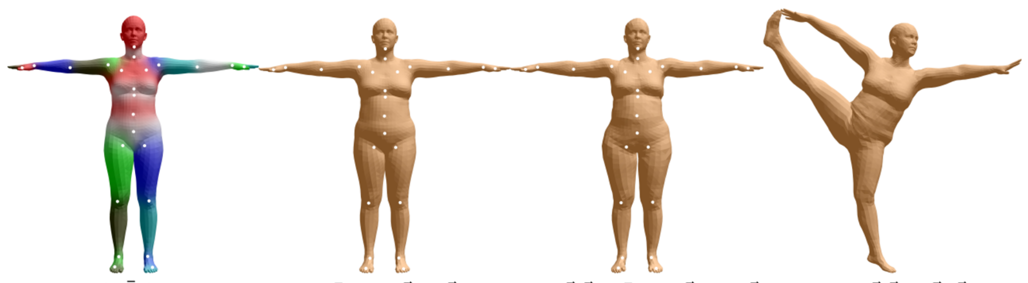
1. **Human modeling: skeleton-based or shape-based**
   Skeleton-based human models aim to predict the 3D joint locations required to pose a predefined skeleton model, also known as a landmark topology. The specific joint definitions and the number of joints may vary across different skeleton models. For instance, OpenPose [22] supports body_25 (25 keypoints), COCO (17 keypoints) [66], and MPI (16 keypoints) [8]. MediaPipe [72] employs a 33-joint skeleton derived from BlazePose [13]. AlphaPose [38] introduces a novel topology comprising 136 keypoints, including 26 for body joints and the remaining keypoints for facial expressions and hand poses. Figure 5.1 shows an illustration of the aforementioned skeleton models. In contrast, shape-based human models, notably the SMPL family, such as SMPL [71], SMPL-X [85], and STAR [82], focus on predicting 3D human body shapes rather than joint locations. An SMPL-based model includes blend weights (W), shape parameters (β), and pose

parameters (θ). Blend weights W encode how a joint rotation affects the position of the vertices, while shape parameters determine body shape, including body physique and scales. Pose parameters (θ) define body pose and shape pose corrective shapes and refine the body shapes according to the current joint rotations. The posing process depicted in Figure 5.2 is consistent across all SMPL-based models, with variations in joint numbers and model training process.



(a) Body_25 [22]. **(b)** COCO [22]. **(c)** MPII [22]. **(d)** BlazePose [13]. **(e)** AlphaPose [38].

Figure 5.1: Skeleton models for human pose estimation.



(a) $\bar{\mathbf{T}}, \mathcal{W}$ **(b)** $\bar{\mathbf{T}} + B_S(\vec{\beta}), J(\vec{\beta})$ **(c)** $T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$ **(d)** $W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W})$

Figure 5.2: SMPL model explanation. (a) Template mesh with blend weights indicated by color and joints shown in white. (b) With identity-driven blend shape contribution only; vertex and joint locations are linear in shape vector. (c) With the addition of pose blend shapes in preparation for the split pose; note the expansion of the hips. (d) Deformed vertices reposed by dual quaternion skinning for the split pose [71].

2. **3D prediction methodology: 2D-to-3D or direct 3D**
   3D pose estimation can be accomplished using two primary approaches: direct estimation of 3D pose and lifting 2D poses derived from existing methods. The prevailing focus of current research lies within the realm of 2D-to-3D prediction, owing to the success of prominent 2D pose estimation techniques, such as OpenPose [22], MediaPipe [72], and AlphaPose [38]. In this approach, 2D poses are elevated to their corresponding 3D counterparts by employing multi-view images and epipolar geometry as illustrated in Figure 5.3. Alternatively, in the case of monocular inputs, temporal information can be leveraged. Compared to direct 3D, the 2D-to-3D lifting task is relatively simpler, as the input data already encompasses reliable 2D pose information. In contrast, direct 3D prediction entails inferring 3D poses directly from images or videos through end-to-end learning. The main advantage of this method is its independence from the performance limitations of any specific 2D pose estimation methods. However, it typically requires the utilization of multi-view inputs to achieve robust performance.
3. **Input data types: monocular or multi-view inputs**
   In recent years, there has been a strong emphasis in research on utilizing monocular input due to its simplicity in managing input data. However, when dealing with monocular inputs, the depth ambiguity problem poses a significant challenge, as there is insufficient information for accurate 3D pose reconstruction. Consequently, in applications where high accuracy is essential, the use of multi-view inputs is preferred, despite the additional effort involved in camera calibration and synchronization.

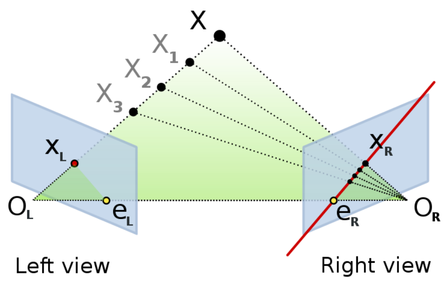


Figure 5.3: Illustration of epipolar geometry [109].

5.1.1. Shaped-based Direct 3D HPE

Shape-based pose estimation using the SMPL model is a common approach for direct 3D human pose estimation, involving the prediction of human model parameters or vertices to reconstruct the 3D human mesh. Table 5.1 provides a summary of the methods discussed in this section.

Due to the success of CNNs in learning visual features, CNNs are widely used as the network backbone [52, 28]. Recently, more works are adapting transformer architecture for visual tasks to learn long-range image features [65]. While some works focus on predicting SMPL shape β and pose θ parameters [52], it is also possible to directly predict vertex positions [65, 28] for pose estimation. To deal with the lack of 3D annotations, the supervision can be a combination of 2D and 3D joint positions [52, 53]. Sequence-to-sequence prediction is present in VIBE [53], which also has a CNNs backbone for frame feature encoding and a Gated Recurrent Unit (GRU) to capture temporal dependencies of SMPL parameters, as shown in Figure 5.4. Adversarial learning can be applied to improve the realism of the predicted pose [52, 53].

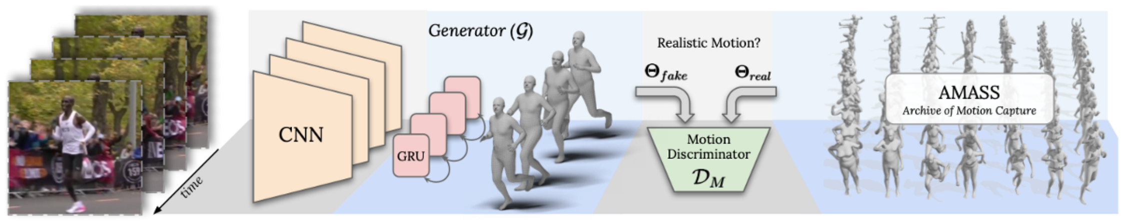

time
CNN
GRU
Generator ($\mathcal{G}$)
Realistic Motion?
Θ_{fake}
Θ_{real}
Motion Discriminator $\mathcal{D}_M$
AMASS
Archive of Motion Capture

Figure 5.4: VIBE architecture [53].

Methods	Year	Input			Highlights	Datasets
HMR [52]	2018	single image	frame2frame	direct 3D	CNN image encoder, weakly-supervised 2D-to-3D, discriminator	Human3.6M, MPI-INF-3DHP
VIBE [53]	2020	monocular video	seq2seq	direct 3D	CNN, GRU, adversarial learning	3DPW, MPI- INF-3DHP, Human3.6M
METRO [65]	2021	single image	frame2frame	direct 3D	CNN, transformer encoder with joint queries and vertex queries	Human3.6M, MPI-INF-3DHP
LVD [28]	2022	single image	frame2frame	direct 3D	Displacements of vertex, hour-glass network, learned human prior	RenderPeople

Table 5.1: Shape-based 3D HPE methods.

5.1.2. Skeleton-based direct 3D HPE

Compared to 2D-to-3D lifting methods, the number of skeleton-based approaches for direct 3D pose estimation is limited. The majority of direct 3D methods employ multi-view inputs. Therefore. effective feature learning from different views for 3D HPE becomes the key to achieving favorable performance.

In [116], a transformer decoder is utilized to effectively model pairwise interactions among all joints from all individuals within a view. Features from multiple views are then aggregated using a projective attention layer. It is also possible to use graph modules for 3D HPE [110], as long as the HPE problem can be interpreted into graph modules properly. For example, the 2D projection of a 3D point is treated as a node, and information from the 2D images is exchanged across views through graph edges as shown in Figure 5.5.

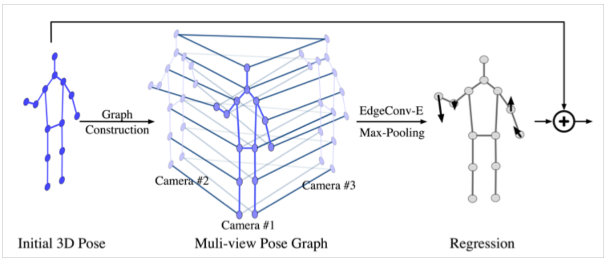


Figure 5.5: Multi-view pose graph proposed in [110].

5.1.3. Skeleton-based 2D to 3D Lifting HPE

State-of-the-art 2D pose estimation methodologies serve as the fundamental building blocks for 2D to 3D Lifting HPE. These approaches aim to leverage 2D poses extracted from multi-view images or a sequence of frames and subsequently lift them to the corresponding 3D pose representations. Among all types of inputs, monocular videos represent a prevalent input modality in this domain due to their ease of processing input data. Table 5.2 summarizes all skeleton-based methods introduced in this section.

CNN-based approaches. Temporal Convolutional Network (TCN) was first proposed in [12] to enable temporal feature learning via CNNs. Because temporal information provides crucial cues for 3D pose reconstruction, TCNs are widely used to process a sequence of 2D joint positions and learn temporal features for reconstructing the 3D pose of the center frame [87, 25]. Some works also propose to consider human anatomy, predicting bone lengths and directions instead of 3D joint positions, for better pose estimation [25].

For the sequence-to-sequence predictions, the task becomes more complex since the network needs to generate realistic movements. The primary objective is to deliver high-quality 2D poses and subsequently employ a network to learn realistic movements based on 2D poses. A concatenation of 2D pose refinement and joint-wise temporal refinement is present in [111], as illustrated in Figure 5.6. Utilization of graphs is also viable since the skeleton topology and the connections of each joint across time can be easily represented as a spatio-temporal graph for human pose modeling [107].

RNN-based approaches. LSTM can be utilized as the temporal feature learner to better estimate 3D human pose along time [48]. To better reduce temporal incoherence and motion jitters, a temporal smoothness constraint is included in the final loss function. However, since recurrent networks

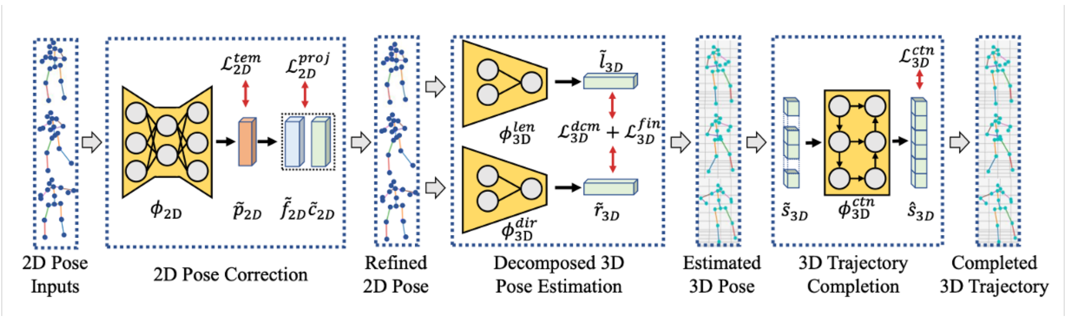


Figure 5.6: The TCN-based sequence-to-sequence 3D HPE framework proposed in [111].

are notoriously unstable when training, there is not much research using recurrent networks as the backbone.

Transformer-based approaches. Transformers can learn spatial and temporal features depending on how the input data is configured. Most of the works comprise one spatial module and one temporal module [120, 61, 117]. The spatial and temporal feature learning can be done in a different order. Poseformer [120] refines spatial features first, [61] refines spatial features last, and [117] refines spatial and temporal feature in iterations, as shown in Figure 5.7. The concept of multiple hypotheses is brought up by MHFormer [63]. Instead of predicting the best 3D pose directly, a combination of transformer-based modules is utilized to find the best 3D pose from multiple possible 3D poses generated from the 2D pose detected in each view.

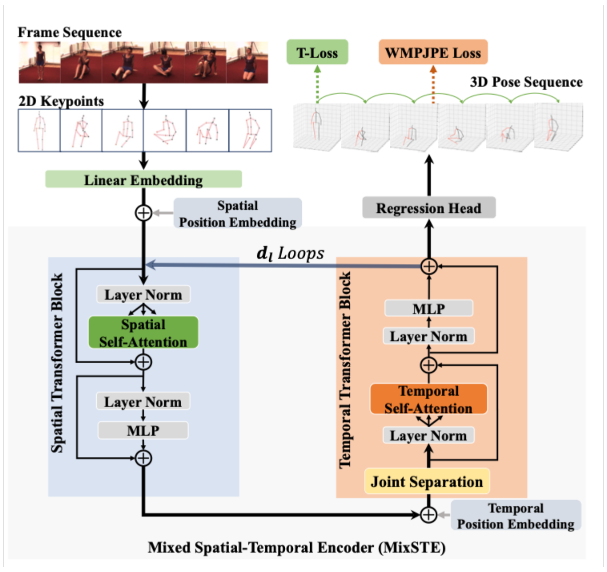


Figure 5.7: The transformer-based iterative spatio-temporal encoder proposed in [117].

Multi-view 3D pose estimation methods offer performance improvements but are hindered by the need for camera calibration. This limitation can be addressed by using a transformer-based multi-view fusion module to refine global features mapped from the 2D poses [96]. It is also possible to eliminate the requirement of using extrinsic camera parameters by predicting both view-dependent pelvis poses and view-independent joint rotations and bone lengths [45]. This approach allows the model to implicitly learn the relative transformation between the cameras.

5.2. 3D HPE with Musculoskeletal Model

Advancements in 3D HPE have facilitated the integration of these techniques with biomechanics models like OpenSim [34]. This fusion enables a comprehensive analysis of human movement and biomechanics, offering insights into kinematics, dynamics, and forces involved in various activities. Table 5.3 summarizes methods introduced in this section.

Building upon the advancements in 2D pose estimation, numerous markerless motion capture methods adopt the 2D-to-3D paradigm, which involves inferring 3D joint positions through triangulation of 2D joint positions detected from multiple views [83, 100]. Pose2Sim [83] proposes a pipeline to predict 3D joint kinematics. The 2D landmark detection backbone can be selected from OpenPose [22], DeepCutLab [75], BlazePose [13], and AlphaPose [38]. During OpenSim IK, the triangulated 3D joints are defined as markers in an OpenSim model. To avoid motion jitters, low-pass filtering is also applied to the 3D joint positions for motion smoothness. OpenCap [100] further introduces an LSTM model to bridge the gap between triangulated 3D landmarks and the 3D joints in an OpenSim model. In comparison to Pose2Sim, OpenCap applies more extensive processing to the detected 2D landmarks, including synchronization, interpolation, and smoothing filtering. After 3D triangulation, RANSAC is employed to

Methods	Year	Input	Highlights			Datasets
Pavlakos et al. [86]	2017	multi-view images	frame2frame	2D-to-3D lifting	ConvNet, 3D pictorial structures, autonomous 3D annotation,	KTH Multiview Football II, Human3.6M
Hossain et al. [48]	2018	monocular video	seq2seq	2D-to-3D lifting	LSTM, temporal smoothness loss	Human3.6M, HumanEva
Pavllo et al. [87]	2019	monocular video	seq2frame	2D-to-3D lifting	Semi-supervised, TCNs, global position, camera intrinsics	Human3.6M, HumanEva
Xu et al. [111]	2020	monocular video	seq2seq	2D-to-3D lifting	TCNs, 2D temporal refinement, decomposed 3D pose estimation, 3D trajectory refinement	Human3.6M, HumanEva
Liu et al. [68]	2020	monocular video	seq2frame	2D-to-3D lifting	Multi-scale TCNs, temporal attention layer	Human3.6M, HumanEva
UGCN [107]	2020	monocular video	seq2seq	2D-to-3D lifting	Spatial-temporal graph, U-shaped Graph Convolution Networks (UGCN), motion loss	Human3.6M, MPI-INF-3DHP
Chen et al. [25]	2021	monocular video	seq2frame	2D-to-3D lifting	Fully-convolutional architecture, attention on 2D visibility scores, per-sequence bone lengths	Human3.6M, MPI-INF-3DHP
Poseformer [120]	2021	monocular video	seq2frame	2D-to-3D lifting	Pure transformer, spatial transformer, temporal transformer	Human3.6M, MPI-INF-3DHP
MvP [116]	2021	multi-view frame	frame2frame	direct 3D	Transformer decoder, camera ray in frame features, feature extraction with 3D to 2D projection on different views	Panoptic
Wu et al. [110]	2021	multi-view images	frame2frame	direct 3D	GNN, multi-view Matching Graph Module (MMG), Center Refinement Graph Module (CRG), Pose Regression Graph Module (PRG)	CMU Panopti, Shelf
MTF-Transform [96]	2022	multi-view video	seq2frame	2D-to-3D lifting	Transformer, Multi-view fusion transformer, temporal fusion transformer	Human3.6M, TotalCapture, KTH Multiview Football II
FLEX [45]	2022	multi-view video	seq2seq	2D-to-3D lifting	Camera extrinsic parameter free, discriminator for motion, global rotation/position, per-sequence bone lengths	Human3.6M
Strided Transformer [61]	2022	monocular video	seq2frame	2D-to-3D lifting	Transformer with stride convolution after vanilla transformer for feature aggregation, full-to-single supervision	Human3.6M, HumanEva
MHFormer [63]	2022	monocular video	seq2frame	2D-to-3D lifting	Transformer, multi-hypothesis	Human3.6M, MPI-INF-3DHP
MixSTE [117]	2022	monocular video	seq2seq	2D-to-3D lifting	Transformer, alternated spatial and temporal transformer blocks	Human3.6M, MPI-INF-3DHP, HumanEva

Table 5.2: Skeleton-based 3D HPE methods.

remove outliers of the 3D keypoints. Also, the video is automatically trimmed based on the confidence scores obtained from the triangulation process.

Both Pose2Sim and OpenCap employ multi-step processing to obtain accurate joint kinematics and body scales for OpenSim models. However, the manual intervention required during these intermediate stages introduces the potential for errors. In order to mitigate this limitation, there is a growing preference for end-to-end frameworks that can simultaneously estimate joint kinematics and body scales. D3KE [15] introduces an end-to-end solution for joint kinematics and body scale estimation as illustrated in Figure 5.8. The proposed method employs CNNs to estimate joint kinematics and body scales for each frame of a monocular video input. Subsequently, a lifting transformer encoder [62] is utilized to refine and aggregate the predictions, producing a 3D pose for the target frame by incorporating temporal information. Experimental evaluations validate the effectiveness of the end-to-end approach, exhibiting superior performance compared to conventional multi-step methods.

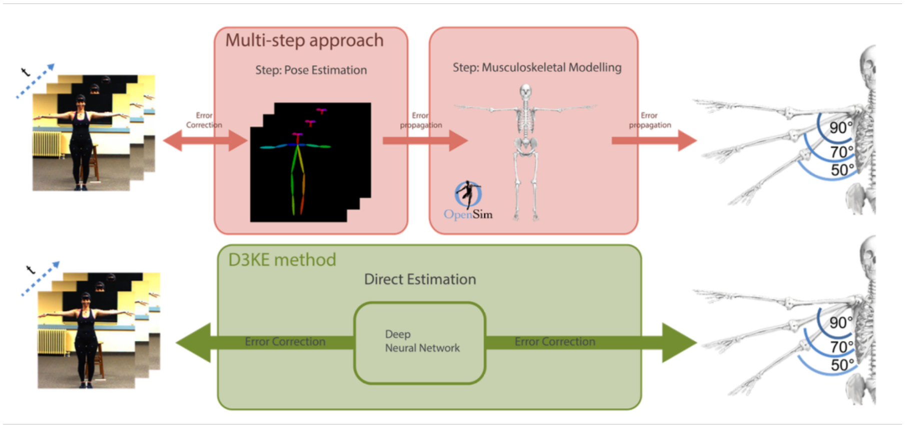


Figure 5.8: The comparison of multi-step and end-to-end methods for 3D HPE based on OpenSim model [15].

Methods	Year	Input		Highlights		Datasets
Pagnon et al. [83]	2022	multi-view video	seq2seq	2D-to-3D lifting	2D keypoints triangulation, refinement with re-projection error and keypoint scores, smoothing filtering on triangulated 3D keypoints	N/A
Uhlrich et al. [100]	2022	multi-view video	seq2seq	2D-to-3D lifting	2D keypoints triangulation, refinement with re-projection error, keypoint scores, RANSAC, smoothing filtering on 2D keypoints, synchronize 2D keypoints, trim low-quality frames	OpenCap
Bittner et al. [15]	2022	monocular video	seq2frame	direct 3D	End-to-end framework, CNNs for spatial prediction, Transformer for temporal smoothing, predict marker positions as auxiliary during training	BMLMovi

Table 5.3: 3D HPE with OpenSim.

5.3. Evaluation Metrics

Mean Per Joint Position Error (MPJPE)

The Mean Per Joint Position Error (MPJPE) is a metric that measures the average Euclidean distance between the predicted 3D joints and the ground truth 3D joints. All positions in MPJPE are relative to the root joint. The equation is shown in Equation (5.1).

$$MPJPE = \frac{1}{N}\sum_{i=1}^{N}\|\hat{\boldsymbol{p}}_i - \boldsymbol{p}_i\|_2, \tag{5.1}$$

, where N is the number of joints, $\hat{\boldsymbol{p}}_i$ and $\boldsymbol{p}_i$ is the predicted and ground truth 3D position of joint i. Note that the positions are first centered on the root joint, such that the joint positions are all relative to the root joint. $\|\cdot\|_2$ denotes the L2-norm, also known as Euclidean distance, between two vectors.

Mean Per Joint Position Error after Procrustes Analysis (PA-MPJPE)

PA-MPJPE not only excludes the error from global translation but also accounts for global rotation and scaling. To achieve this, a similarity transform obtained through Procrustes Analysis (PA) is applied to the predicted pose prior to calculating MPJPE. This ensures that the predicted pose is better aligned with the ground truth pose in terms of global translation, rotation, and scaling. The equation is shown in Equation (5.2).

$$\begin{gathered} PA\text{-}MPJPE = \frac{1}{N}\sum_{i=1}^{N}\|\hat{\boldsymbol{p}}_i^{PA} - \boldsymbol{p}_i\|_2, \\ \hat{\boldsymbol{p}}_i^{PA} = M \times \hat{\boldsymbol{p}}_i \times s + \boldsymbol{t} \end{gathered} \tag{5.2}$$

, where N is the number of joints, $\hat{\boldsymbol{p}}_i$ and $\boldsymbol{p}_i$ are the predicted and ground truth joint positions relative to the root joint. $\hat{\boldsymbol{p}}_i^{PA}$ is the joint position after being rotated, scaled, and translated with a similarity transform matrix M derived from Procrustes analysis. Note that $M \in \mathbb{R}^{3x3}$, $\boldsymbol{t} \in \mathbb{R}^{3x1}$, and s is a scalar. The notation $\|\cdot\|_2$ denotes the L2-norm between two vectors.

Mean Per Joint Velocity Error (MPJVE)

For evaluating the smoothness of predicted human pose sequences, the Mean Per Joint Velocity Error (MPJVE) measures the average error between the predicted joint velocities and the ground truth joint velocities, providing an assessment of the temporal coherence and smoothness of the predicted pose sequence. The equation is shown in Equation (5.3).

$$MPJVE = \frac{1}{(T-1)N} \sum_{t=2}^{T} \sum_{i=1}^{N} \|\hat{\boldsymbol{v}}_{i,t} - \boldsymbol{v}_{i,t}\|_2 \tag{5.3}$$

, where N is the number of joints, T is the total number of frames, and $\hat{\boldsymbol{v}}_{i,t}$ and $\boldsymbol{v}_{i,t}$ are the predicted and ground truth joint displacements in frame $t-1$, respectively. The notation $\|\cdot\|_2$ denotes the L2-norm between two vectors. Note that MPJVE can also be calculated after applying Procrustes analysis.

Point to Vertex Error (PVE)

In the context of shape-based human modeling, the Percentage of Vertex Error (PVE) metric assesses the similarity between the predicted mesh and the ground truth. Both the predicted and ground truth meshes undergo alignment by aligning their respective root vertices. The equation is shown in Equation (5.4).

$$PVE = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{p}_i - \boldsymbol{v}_i\|_2 \tag{5.4}$$

, where N represents the number of points/vertices, $\boldsymbol{p}_i$ is the position of the i-th point, and $\boldsymbol{v}_i$ denotes the position of the corresponding i-th vertex. The notation $\|\cdot\|_2$ denotes the L2-norm between two vectors.

5.4. Datasets

In this section, we provide Table 5.4 to summarize some popular datasets for 3D HPE.

Dataset	Year	Subjects	Actions	Frames	Cameras	MoCap	Notes
HumanEva [97]	2010	4	5	~80k	7	v	Studio, Natural clothes
Human3.6M [49]	2013	11	11	3.6M	4	v	Studio, Natural clothes
CMU Panoptic [51]	2015	~8	>10	1.1M	31		Studio, Natural clothes, Multi-person, social interaction
TotalCapture [99]	2017	5	4	~1.9M	8	v	Studio , Suit
MPI-INF-3DH [76]	2017	8	8	>1.3M	14		Studio, Natural clothes, Augmented backgrounds
SURREAL [101]	2017	145	23	>6M	1		Synthetic data
3DPW [74]	2018	7	n/a	~51k	1		In-the-wild, Natural clothes, Moving camera
BMLMovi [43]	2020	90	21	3.6M	4	v	Studio, Suit and normal clothes
OpenCap [100]	2022	10	8	100k	5	v	Studio, Minimum clothes, Asymmetrical movements, OpenSim annotation

Table 5.4: Datasets for 3D human pose estimation.

6 Musculoskeletal Model – OpenSim

OpenSim [34] is an open-source software platform that has been specifically developed to facilitate the creation and simulation of musculoskeletal models for human movement analysis. This software enables users to construct detailed models of the musculoskeletal system, encompassing elements such as bones, muscles, tendons, and ligaments, and subsequently simulate the corresponding movements of the human body. Furthermore, OpenSim provides an extensive array of tools and functionalities for the analysis of kinematics, kinetics, and muscle forces implicated in movement, as well as for the optimization of movement patterns and the development of rehabilitation protocols. Owing to its comprehensive features, OpenSim has gained widespread utilization across various disciplines, encompassing a diverse range of studies for human movement analysis, such as gait analysis, sports performance assessment, and planning for orthopedic surgery.

In the following sections, we will present an overview of the essential elements that constitute an OpenSim model. Additionally, we will discuss the scaling and inverse kinematics tool, and delve into the joint kinematics forward calculation in OpenSim.

6.1. Model Components

Body

The bodies in an OpenSim model are defined as the segments in the human body. In our generic model [91], there are 22 bodies, including pelvis, femur_r, tibia_r, patella_r, talus_r, calcn_r, toes_r, femur_l, tibia_l, patella_l, talus_l, calcn_l, toes_l, torso, humerus_r, ulna_r, radius_r, hand_r, humerus_l, ulna_l, radius_l, and hand_l. An illustration of the generic model and the corresponding bodies and joints is shown in Figure 6.1. The 22 bodies are connected via joints, which control the position and rotation of each body.

Body Segment Scales

In OpenSim, the size of each body is defined by 3D scaling factors, a total of 22×3 values, to the same body in a referenced model.

Joints

OpenSim joints serve as pivotal components in establishing connections among all bodies in a model. Each joint is defined by a parent frame and a child frame. Both consist of offset frames representing a fixed 3D translation relative to a body. Changes in the position and rotation of a joint directly alter the position and rotation of the associated child offset frame. Since the parent offset frame of the child joint is tied with the child offset frame of the current joint, the position and rotation of the child joint will change accordingly. Following the hierarchical structure of the skeleton, the joint movements will be propagated to bodies at lower hierarchical levels. The generic model we used [91] encompasses a total of 22 joints, and the corresponding bodies associated with each joint are visually represented in Figure 6.1.

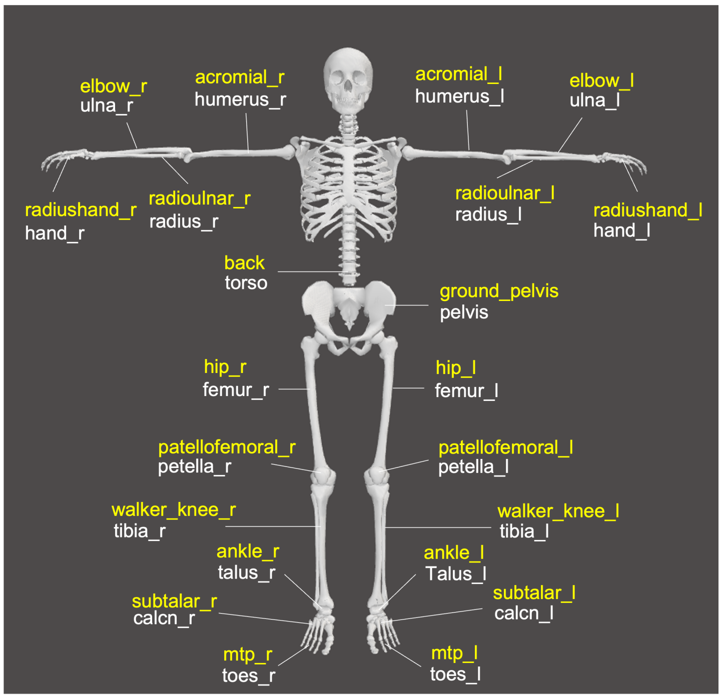


Figure 6.1: The OpenSim model proposed in [91] and its joint-body pairs. The joints are marked in yellow. Bodies are marked in white.

Coordinates

Coordinates, commonly referred to as joint angles, control the joint rotation and translation through user-defined mapping functions. Our generic model comprises a total of 39 coordinates. The joints and their corresponding coordinates are presented in Table 6.1.

Joint	Coordinate
ground_pelvis	pelvis_tilt, pelvis_list, pelvis_rotation, pelvis_tx, pelvis_ty, pelvis_tz
hip_r/l	hip_flextion_r/l, hip_adduction_r/l, hip_rotation_r/l
patellofemoral_r/l	knee_angle_beta_r/l
walker_knee_r/l	knee_angle_r/l
ankle_r/l	ankle_angle_r/l
subtalar_r/l	subtalar_angle_r/l
mtp_r/l	mtp_angle_r/l
back	lumbar_extension, lumbar_bending, lumbar_rotation
acromial_r/l	arm_flex_r/l, arm_add_r/l, arm_rot_r/l
elbow_r/l	elbow_flex_r/l
radioulnar_r/l	pro_sup_r/l
radiushand_r/l	wrist_flex_r/l, wrist_dev_r/l

Table 6.1: Joint names and their corresponding coordinates.

6.2. Scaling Tool and Inverse Kinematics Tool

OpenSim offers functionalities for performing body scale computation and inverse kinematics analysis. Users can specify the markers that are used to derive the segment scales of each body. On the other hand, inverse kinematics involves the estimation of joint angles and positions using measured marker trajectories. Typically, the scaling tool is employed initially to obtain the subject's body scales. Subsequently, the scaled model and the marker trajectories are utilized in conjunction with the inverse kinematics tool.

6.3. Forward Kinematics

The 3D joint kinematics, including positions and rotations, can be derived given the coordinates and body segment scales. This computation involves a series of matrix multiplications. We will provide an example of knee movement to explain the computation, as depicted in Figure 6.2.

In Figure 6.2, red represents the transformation matrix between joints (marked in yellow) and body (marked in white). The joint $walker\_knee\_l$ consists of a parent offset frame $kneeP$ and a child offset frame $kneeC$. Both offset frames are defined by a 3D translation of their parent body. The parent body of $kneeP$ is $femur\_l$, and X_{femur_kneeP} is a transformation matrix with the defined 3D offset. Likewise, the parent body of $kneeC$ is $tibia_l$, and X_{tibia_kneeC} is the defined 3D offset. The positions and rotations of body $femur\_l$ and body $tibia\_l$ with respect to the ground are represented by transformation matrices X_{G_femur} and X_{G_tibia}, respectively. Finally, the transformation from $kneeP$ to $kneeC$ is defined by a transformation matrix X_{kneeP_kneeC}. Among these matrices, X_{femur_kneeP} and X_{tibia_kneeC} are defined in the model and remain fixed, while X_{G_femur}, X_{G_tibia}, and X_{kneeP_kneeC} change as the pose changes.

The transformation matrix X_{kneeP_kneeC}, which determines the movement of the joint $walker\_knee\_l$, is derived by coordinate $knee\_angle\_l$. The computation from coordinate $knee\_angle\_l$ to X_{kneeP_kneeC} is as follows:

$$\begin{aligned} X_{kneeP\_kneeC} &= [R_{kneeP\_kneeC}|T_{kneeP\_kneeC}], \\ R_{kneeP\_kneeC} &= euler\_to\_matrix(f_{r0}(C_{17}), f_{r1}(C_{17}), f_{r2}(C_{17}), euler\_mode), \\ T_{kneeP\_kneeC} &= [f_{tx}(C_{17}, scale_{femur}), f_{ty}(C_{17}, scale_{femur}), f_{tz}(C_{17}, scale_{femur})] \end{aligned}$$

, where C_{17} represents the value of coordinate $knee\_angle\_l$. The functions $f_{ri}(\cdot)$, where $i = 0, 1, 2$, are pre-defined mapping functions that convert the coordinate value to the 3D rotation angles around different axes. Similarly, the functions $f_{ti}(\cdot)$, where $i = x, y, z$, are pre-defined mapping functions that determine the translation between $kneeP$ and $kneeC$ given the coordinate value. The mapping functions can be defined by arbitrary functions, for instance, spline functions, linear functions, constants, and so on. The function $euler\_to\_matrix(\cdot)$ is used to convert the Euler angles to rotation matrices. The order of rotation axes, denoted by $euler\_mode$, is also specified by the users. Additionally, it should be noted that the translation vector T_{kneeP_kneeC} is influenced by the body segment scales of the parent offset frame's parent body.

After obtaining X_{kneeP_kneeC}, we can derive the transformation matrix of the child joint $kneeC$ and body $tibia\_l$ with respect to ground, using X_{G_femur} and X_{femur_kneeP}. The computation is given by:

$$\begin{aligned} X_{G\_kneeC} &= X_{G\_femur} \cdot X_{femur\_kneeP} \cdot X_{kneeP\_kneeC} \\ X_{G\_tibia} &= X_{G\_kneeC} \cdot (X_{tibia\_kneeC})^{-1} \end{aligned}$$

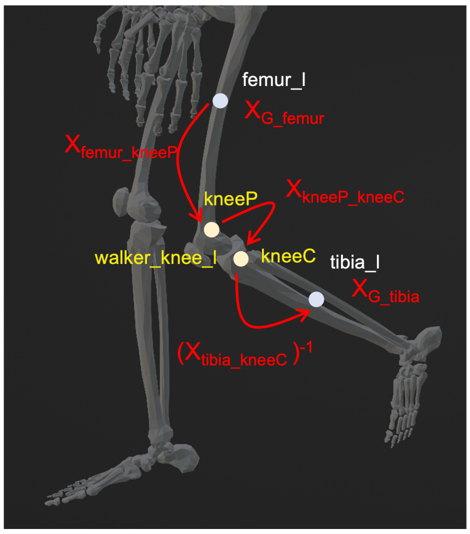


Figure 6.2: Illustration of joint kinematics forward computation. The joints are marked in yellow. Bodies are marked in white. Transformation matrices are in red.

References

[1] URL: `https://www.digitalocean.com/community/tutorials/how-to-build-a-neural-network-to-recognize-handwritten-digits-with-tensorflow`.

[2] URL: `https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9`.

[3] URL: `https:https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`.

[4] URL: `https://d2l.ai/chapter_recurrent-modern/lstm.html`.

[5] URL: `https://d2l.ai/chapter_recurrent-modern/gru.html`.

[6] URL: `https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a`.

[7] URL: `https://openai.com/blog/chatgpt`.

[8] Mykhaylo Andriluka et al. “2d human pose estimation: New benchmark and state of the art analysis”. In: *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 2014, pp. 3686–3693.

[9] Antreas Antoniou, Amos Storkey, and Harrison Edwards. “Data augmentation generative adversarial networks”. In: *arXiv preprint arXiv:1711.04340* (2017).

[10] Anurag Arnab et al. “ViViT: A Video Vision Transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 6836–6846.

[11] Rahma Atallah and Amjed Al-Mousa. “Heart disease detection using machine learning majority voting ensemble method”. In: *2019 2nd international conference on new trends in computing sciences (ictcs)*. IEEE. 2019, pp. 1–6.

[12] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”. In: *arXiv preprint arXiv:1803.01271* (2018).

[13] Valentin Bazarevsky et al. “Blazepose: On-device real-time body pose tracking”. In: *arXiv preprint arXiv:2006.10204* (2020).

[14] Shai Ben-David et al. “Analysis of representations for domain adaptation”. In: *Advances in neural information processing systems* 19 (2006).

[15] Marian Bittner et al. “Towards Single Camera Human 3D-Kinematics”. In: *Sensors* 23.1 (2022), p. 341.

[16] Michael J. Black et al. “BEDLAM: A Synthetic Dataset of Bodies Exhibiting Detailed Lifelike Animated Motion”. In: *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. June 2023.

[17] Gilles Blanchard, Gyemin Lee, and Clayton Scott. “Generalizing from several related classification tasks to a new unlabeled sample”. In: *Advances in neural information processing systems* 24 (2011).

[18] Pratik Prabhanjan Brahma, Dapeng Wu, and Yiyuan She. “Why deep learning works: A manifold disentanglement perspective”. In: *IEEE transactions on neural networks and learning systems* 27.10 (2015), pp. 1997–2008.

[19] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32.

[20] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[21] Hu Cao et al. “Swin-unet: Unet-like pure transformer for medical image segmentation”. In: *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*. Springer. 2023, pp. 205–218.

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

[23] Chen Chen et al. “Learning to see in the dark”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3291–3300.

[24] Chun-Fu (Richard) Chen, Quanfu Fan, and Rameswar Panda. “CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 357–366.

[25] Tianlang Chen et al. “Anatomy-aware 3d human pose estimation with bone-based pose decomposition”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 32.1 (2021), pp. 198–209.

[26] Wuyang Chen et al. “Contrastive syn-to-real generalization”. In: *arXiv preprint arXiv:2104.02290* (2021).

[27] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[28] Enric Corona et al. “Learned vertex descent: a new direction for 3D human model fitting”. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*. Springer. 2022, pp. 146–165.

[29] Gabriela Csurka. “Domain adaptation for visual applications: A comprehensive survey”. In: *arXiv preprint arXiv:1702.05374* (2017).

[30] Ekin D Cubuk et al. “Autoaugment: Learning augmentation policies from data”. In: *arXiv preprint arXiv:1805.09501* (2018).

[31] Ekin D Cubuk et al. “Autoaugment: Learning augmentation strategies from data”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 113–123.

[32] Zhiyong Cui et al. “Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction”. In: *arXiv preprint arXiv:1801.02143* (2018).

[33] Antonio D’Innocente and Barbara Caputo. “Domain generalization with domain-specific aggregation modules”. In: *Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings 40*. Springer. 2019, pp. 187–198.

[34] 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.

[35] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[36] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[37] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[38] Hao-Shu Fang et al. “AlphaPose: Whole-Body Regional Multi-Person Pose Estimation and Tracking in Real-Time”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[39] *Feedforward Neural Networks*. URL: `https://brilliant.org/wiki/feedforward-neural-networks/`.

[40] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[41] Yoav Freund, Robert E Schapire, et al. “Experiments with a new boosting algorithm”. In: *icml*. Vol. 96. Citeseer. 1996, pp. 148–156.

[42] Yaroslav Ganin and Victor Lempitsky. “Unsupervised domain adaptation by backpropagation”. In: *International conference on machine learning*. PMLR. 2015, pp. 1180–1189.

[43] Saeed Ghorbani et al. “MoVi: A large multi-purpose human motion and video dataset”. In: *Plos one* 16.6 (2021), e0253157.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[45] Brian Gordon et al. “FLEX: Extrinsic Parameters-free Multi-view 3D Human Motion Reconstruction”. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*. Springer. 2022, pp. 176–196.

[46] Pengcheng He et al. “Deberta: Decoding-enhanced bert with disentangled attention”. In: *arXiv preprint arXiv:2006.03654* (2020).

[47] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[48] Mir Rayat Imtiaz Hossain and James J Little. “Exploiting temporal information for 3d human pose estimation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 68–84.

[49] Catalin Ionescu et al. “Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments”. In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2013), pp. 1325–1339.

[50] NS Johnson et al. “Machine Learning for Materials Developments in Metals Additive Manufacturing”. In: *arXiv preprint arXiv:2005.05235* (2020).

[51] Hanbyul Joo et al. “Panoptic studio: A massively multiview system for social motion capture”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3334–3342.

[52] Angjoo Kanazawa et al. “End-to-end recovery of human shape and pose”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7122–7131.

[53] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. “Vibe: Video inference for human body pose and shape estimation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 5253–5263.

[54] Ruho Kondo et al. “Flow-based image-to-image translation with feature disentanglement”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[56] Zhenzhong Lan et al. “Albert: A lite bert for self-supervised learning of language representations”. In: *arXiv preprint arXiv:1909.11942* (2019).

[57] Sohyun Lee, Taeyoung Son, and Suha Kwak. “Fifo: Learning fog-invariant features for foggy scene segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18911–18921.

[58] Alexander Lehner et al. “3D-VField: Adversarial Augmentation of Point Clouds for Domain Generalization in 3D Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17295–17304.

[59] Da Li et al. “Learning to generalize: Meta-learning for domain generalization”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[60] Yu-Jhe Li et al. “Cross-dataset person re-identification via unsupervised pose disentanglement and adaptation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 7919–7929.

[61] Wenhao Li et al. “Exploiting temporal contexts with strided transformer for 3d human pose estimation”. In: *IEEE Transactions on Multimedia* (2022).

[62] Wenhao Li et al. “Lifting transformer for 3d human pose estimation in video”. In: *arXiv preprint arXiv:2103.14304* 2 (2021).

[63] Wenhao Li et al. “Mhformer: Multi-hypothesis transformer for 3d human pose estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13147–13156.

[64] Henry W Lin, Max Tegmark, and David Rolnick. “Why does deep and cheap learning work so well?” In: *Journal of Statistical Physics* 168 (2017), pp. 1223–1247.

[65] Kevin Lin, Lijuan Wang, and Zicheng Liu. “End-to-end human pose and mesh reconstruction with transformers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 1954–1963.

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

[67] Chang Liu et al. “Learning to learn across diverse data biases in deep face recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4072–4082.

[68] Ruixu Liu et al. “Attention mechanism exploits temporal contexts: Real-time 3d human pose reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5064–5073.

[69] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[70] Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 10012–10022.

[71] Matthew Loper et al. “SMPL: A Skinned Multi-Person Linear Model”. In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6 (Oct. 2015), 248:1–248:16.

[72] Camillo Lugaresi et al. *MediaPipe: A Framework for Building Perception Pipelines*. 2019. arXiv: `1906.08172 [cs.DC]`.

[73] Junhua Mao et al. “Deep captioning with multimodal recurrent neural networks (m-rnn)”. In: *arXiv preprint arXiv:1412.6632* (2014).

[74] Timo von Marcard et al. “Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera”. In: *European Conference on Computer Vision (ECCV)*. Sept. 2018.

[75] Alexander Mathis et al. “DeepLabCut: markerless pose estimation of user-defined body parts with deep learning”. In: *Nature neuroscience* 21.9 (2018), pp. 1281–1289.

[76] Dushyant Mehta et al. “Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision”. In: *3D Vision (3DV), 2017 Fifth International Conference on*. IEEE. 2017. DOI: `10.1109/3dv.2017.00064`. URL: `http://gvv.mpi-inf.mpg.de/3dhp_dataset`.

[77] Ishan Misra and Laurens van der Maaten. “Self-supervised learning of pretext-invariant representations”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6707–6717.

[78] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. “Recurrent models of visual attention”. In: *Advances in neural information processing systems* 27 (2014).

[79] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. “Methods for interpreting and understanding deep neural networks”. In: *Digital signal processing* 73 (2018), pp. 1–15.

[80] Guido F Montufar et al. “On the number of linear regions of deep neural networks”. In: *Advances in neural information processing systems* 27 (2014).

[81] Alejandro Newell, Kaiyu Yang, and Jia Deng. “Stacked hourglass networks for human pose estimation”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*. Springer. 2016, pp. 483–499.

[82] Ahmed A A Osman, Timo Bolkart, and Michael J. Black. “STAR: A Sparse Trained Articulated Human Body Regressor”. In: *European Conference on Computer Vision (ECCV)*. 2020, pp. 598–613. URL: `https://star.is.tue.mpg.de`.

[83] David Pagnon, Mathieu Domalain, and Lionel Reveret. “Pose2Sim: An open-source Python package for multiview markerless kinematics”. In: *Journal of Open Source Software* (2022). DOI: `10.21105/joss.04362`. URL: `https://joss.theoj.org/papers/10.21105/joss.04362`.

[84] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[85] Georgios Pavlakos et al. "Expressive Body Capture: 3D Hands, Face, and Body from a Single Image". In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10975–10985.

[86] Georgios Pavlakos et al. "Harvesting multiple views for marker-less 3d human pose annotations". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6988–6997.

[87] Dario Pavllo et al. "3d human pose estimation in video with temporal convolutions and semi-supervised training". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 7753–7762.

[88] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[89] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[90] Maithra Raghu et al. "Transfusion: Understanding transfer learning for medical imaging". In: *Advances in neural information processing systems* 32 (2019).

[91] Apoorva Rajagopal et al. "Full-body musculoskeletal model for muscle-driven simulation of human gait". In: *IEEE transactions on biomedical engineering* 63.10 (2016), pp. 2068–2079.

[92] Benjamin Recht et al. "Do imagenet classifiers generalize to imagenet?" In: *International conference on machine learning*. PMLR. 2019, pp. 5389–5400.

[93] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[94] Nirat Saini, Khoi Pham, and Abhinav Shrivastava. "Disentangling visual embeddings for attributes and objects". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13658–13667.

[95] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1 (2019), pp. 1–48.

[96] Hui Shuai, Lele Wu, and Qingshan Liu. "Adaptive multi-view and temporal fusing transformer for 3d human pose estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[97] Leonid Sigal, Alexandru O Balan, and Michael J Black. "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion". In: *International journal of computer vision* 87.1-2 (2010), p. 4.

[98] Flood Sung et al. "Learning to compare: Relation network for few-shot learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1199–1208.

[99] Matt Trumble et al. "Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors". In: *2017 British Machine Vision Conference (BMVC)*. 2017.

[100] Scott D Uhlrich et al. "OpenCap: 3D human movement dynamics from smartphone videos". In: *bioRxiv* (2022), pp. 2022–07.

[101] Gül Varol et al. "Learning from Synthetic Humans". In: *CVPR*. 2017.

[102] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[103] Riccardo Volpi et al. "Generalizing to unseen domains via adversarial data augmentation". In: *Advances in neural information processing systems* 31 (2018).

[104] Fan Wang et al. "Exploring domain-invariant parameters for source free domain adaptation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7151–7160.

[105] Haotao Wang et al. “Augmax: Adversarial composition of random augmentations for robust training”. In: *Advances in neural information processing systems* 34 (2021), pp. 237–250.

[106] Jindong Wang et al. “Generalizing to unseen domains: A survey on domain generalization”. In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[107] Jingbo Wang et al. “Motion guided 3d pose estimation from videos”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*. Springer. 2020, pp. 764–780.

[108] Wenhai Wang et al. “Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction Without Convolutions”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 568–578.

[109] Wikipedia. *Epipolar geometry — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Epipolar%20geometry&oldid=1125486185`. [Online; accessed 07-April-2023]. 2023.

[110] Size Wu et al. “Graph-based 3d multi-person pose estimation using multi-view images”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 11148–11157.

[111] Jingwei Xu et al. “Deep kinematics analysis for monocular 3d human pose estimation”. In: *Proceedings of the IEEE/CVF Conference on computer vision and Pattern recognition*. 2020, pp. 899–908.

[112] Hao Xue, Du Q Huynh, and Mark Reynolds. “SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1186–1194.

[113] Syed Waqas Zamir et al. “Multi-Stage Progressive Image Restoration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 14821–14831.

[114] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer. 2014, pp. 818–833.

[115] Hanlin Zhang et al. “Towards principled disentanglement for domain generalization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8024–8034.

[116] Jianfeng Zhang et al. “Direct multi-view multi-person 3d pose estimation”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 13153–13164.

[117] Jinlu Zhang et al. “Mixste: Seq2seq mixed spatio-temporal encoder for 3d human pose estimation in video”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13232–13242.

[118] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. *Hierarchical Recurrent Neural Network for Video Summarization*. 2019. arXiv: `1904.12251 [cs.CV]`.

[119] Yiru Zhao et al. “Attribute-driven feature disentangling and temporal aggregation for video person re-identification”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4913–4922.

[120] Ce Zheng et al. “3d human pose estimation with spatial and temporal transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 11656–11665.

[121] Kaiyang Zhou et al. “Domain generalization: A survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[122] Fuzhen Zhuang et al. “A comprehensive survey on transfer learning”. In: *Proceedings of the IEEE* 109.1 (2020), pp. 43–76.