TUM School of Computation, Information and Technology
Technical University of Munich

Master's Thesis in Data Engineering and Analytics

Exploiting Spatio-Temporal Relationships for 3D Pose with Graph Neural Networks

Ausnutzung räumlich-zeitlicher Beziehungen für die 3D-Pose des Menschen mit graphneuronalen Netzen

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Soubarna Banik, M.Sc.

Author Edvard Avagyan

Date June 15, 2023 in Munich

Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Munich, June 15, 2023 (Edvard Avagyan)

Abstract

Despite the abundance of use-cases demanding motion of a person in 3D space, most 3D human pose estimation models opt to reconstruct 3D pose of a person only from a single frame of input. Thus, the temporal continuity and smoothness of videos are not modeled in these approaches leading to undesirable noise in the reconstructed 3D motion. This thesis focuses on the development of an efficient method for reconstructing the motion of a single person in 3D space by leveraging spatial-temporal relationships in the input 2D joint locations obtained from videos. Two neural networks, named *TGraphNet* and *TGraphNet (traj)* were developed to estimate the 3D pose and motion of individuals. Central to both models is the exploitation of temporal motion of joints, and spatial inter-joint relationships in the skeleton through multi-scale temporal convolutions (TCN) and spatio-temporal graph convolutions (STGCN). Smooth and realistic 3D pose sequence is obtained by incorporating natural skeletal structure and temporal continuity constraints in the model. Furthermore, this thesis explores an extension of TGraphNet, TGraphNet (traj), which not only estimates root relative poses but also predicts the global position of the skeleton in 3D space to achieve a complete motion of the person. TGraphNet (traj) addresses depth ambiguity by exploiting implicit depth cues such as size and scale of body parts present in the input sequence. The evaluation of TGraphNet on multiple benchmark datasets demonstrates that it achieves performance close to the state of the art while requiring fewer computational resources. The model shows the ability to reconstruct smooth motion with low velocity error and benefits from an increased receptive field and adaptive adjacency matrices. However, it exhibits limitations in adapting to camera setups different from the training dataset. TGraphNet (traj) enriches the predicted poses by estimating the global trajectory, but it struggles with depth estimation in scenarios beyond the training camera setup. These limitations highlight the challenges of models trained on specific camera setups highlighting the fact that further improvements are needed for global trajectory prediction and generalization across different camera setups.

Zusammenfassung

Trotz der Vielzahl von Anwendungsfällen, die eine Bewegung einer Person im 3D-Raum erfordern, entscheiden sich die meisten Modelle zur 3D-Menschengelenkpositionsschätzung dafür, die 3D-Pose einer Person nur aus einem einzelnen Bild wiederherzustellen. Dadurch werden die zeitliche Kontinuität und die Glattheit von Videos in diesen Ansätzen nicht modelliert, was zu unerwünschtem Rauschen in der rekonstruierten 3D-Bewegung führt. Diese Arbeit konzentriert sich auf die Entwicklung einer effizienten Methode zur Rekonstruktion der Bewegung einer einzelnen Person im 3D-Raum unter Ausnutzung räumlich-zeitlicher Beziehungen in den aus Videos gewonnenen 2D-Gelenkpositionen. Es wurden zwei neuronale Netzwerke namens *TGraphNet* und *TGraphNet (traj)* entwickelt, um die 3D-Pose und Bewegung von Individuen abzuschätzen. Zentral für beide Modelle ist die Nutzung der zeitlichen Bewegung der Gelenke und der räumlichen Beziehungen zwischen den Gelenken im Skelett durch mehrskalige zeitliche Faltungen (TCN) und räum-zeitliche Graphfaltungen (STGCN). Eine glatte und realistische 3D-Pose-Sequenz wird durch die Einbeziehung der natürlichen Skelettstruktur und der zeitlichen Kontinuitätsbedingungen im Modell erreicht. Darüber hinaus untersucht diese Arbeit eine Erweiterung von TGraphNet, TGraphNet (traj), das nicht nur die relative Position der Wurzelgelenke schätzt, sondern auch die globale Position des Skeletts im 3D-Raum vorhersagt, um eine vollständige Bewegung der Person zu erreichen. TGraphNet (traj) begegnet der Tiefneunklarheit, indem es implizite Tiefenhinweise wie Größe und Skalierung der Körperteile in der Eingabesequenz nutzt. Die Evaluation von TGraphNet anhand mehrerer Benchmark-Datensätze zeigt, dass es eine Leistung erreicht, die nahe an den State-of-the-Art-Methoden liegt und dabei weniger Rechenressourcen erfordert. Das Modell ist in der Lage, eine glatte Bewegung mit geringem Geschwindigkeitsfehler wiederherzustellen und profitiert von einem erweiterten Rezeptionsfeld und adaptiven Adjazenzmatrizen. Es zeigt jedoch Einschränkungen bei der Anpassung an Kameraeinstellungen, die sich von den Trainingsdaten unterscheiden. TGraphNet (traj) verbessert die vorhergesagten Posen durch die Schätzung der globalen Trajektorie, hat jedoch Schwierigkeiten bei der Tiefenschätzung in Szenarien, die über das Trainingskamera-Setup hinausgehen. Diese Einschränkungen verdeutlichen die Herausforderungen von Modellen, die auf spezifischen Kameraeinstellungen trainiert wurden, und zeigen, dass weitere Verbesserungen für die Vorhersage der globalen Trajektorie und die Verallgemeinerung über verschiedene Kameraeinstellungen erforderlich sind.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Outline of the Thesis . . . 2

2 Foundations and Background **5**
2.1 Image Formation . . . 5
2.1.1 Perspective Model . . . 5
2.1.2 Camera Intrinsic Parameters . . . 6
2.1.3 Sequence of Images . . . 6
2.2 Graph Convolutional Neural Networks . . . 7
2.2.1 Motivation of Graph Neural Networks . . . 8
2.2.2 Graph Operations . . . 8
2.2.3 Types of Graph Convolutional Networks . . . 9
2.2.4 Extensions to Graphs and Graph Convolutional Networks . . . 11
2.3 Transformers . . . 13
2.4 Human Pose Estimation . . . 13
2.4.1 Human Body Models . . . 14
2.4.2 Categories of 3D Human Pose Estimation . . . 14
2.4.3 Challenges in 3D Human Pose Estimation . . . 16

3 Related Work **19**
3.1 Single Frame Models . . . 19
3.1.1 GCN Based Methods . . . 19
3.1.2 Transformer Based Methods . . . 22
3.1.3 Summary . . . 22
3.2 Multi Frame Models . . . 23
3.2.1 Spatio-Temporal Convolution Based Methods . . . 23
3.2.2 Transformer Based Methods . . . 25
3.2.3 Summary . . . 28

4 Methodology **31**
4.1 Overview of Spatial-Temporal GCN Based Approach . . . 32
4.2 Formation of The Spatio-Temporal Graph and Feature Extraction . . . 33
4.3 Adjacency Matrix and The Neighborhood Groups . . . 34
4.3.1 Spatial Connections . . . 35
4.3.2 Temporal Connections . . . 35
4.4 Network Modules . . . 36
4.5 Loss Functions . . . 37

5 Experimental Results **41**
5.1 Experimental Setup . . . 41

5.1.1 Datasets . . . 41
5.1.2 Evaluation Metrics . . . 42
5.1.3 Implementation Details . . . 42
5.2 Root Relative 3D Human Pose Estimation . . . 44
5.2.1 Quantitative Evaluation . . . 44
5.2.2 Qualitative Analysis . . . 50
5.2.3 Ablation Study . . . 53
5.2.4 Summary . . . 56
5.3 Trajectory Estimation . . . 56
5.3.1 Quantitative Evaluation . . . 57
5.3.2 Qualitative Evaluation . . . 58
5.3.3 Ablation Study . . . 61
5.3.4 Summary . . . 63

6 Conclusion and Future Work **65**
6.1 Conclusion and Discussion . . . 65
6.2 Future Work . . . 66

A PW3D and Human3.6M Scene Comparison **67**

Bibliography **69**

List of Figures

1.1 Comparison of motion with single-frame and multi-frame models 1

2.1 The pinhole camera model 5
2.2 Demonstration of single frame and sequence to sequence paradigms 7
2.3 2D Grid convolution versus a graph neural network 8
2.4 Common human body models 15
2.5 3D Human pose estimation variants 15

3.1 Graph convolution operation adopted in SemGCN [Zha+19b] 19
3.2 Weight sharing variants in GCNs 20
3.3 Skeletal pooling and skeletal unpooling operations 21
3.4 Visualisation of dilated convolutions in VideoPose3D [Pav+19] 23
3.5 Semi-supervised 3D pose estimation with reprojection loss and a trajectory model 23
3.7 Spatio-temporal graph convolutional network used by [Cai+19b] 24
3.6 Demonstration of the division of joints into neighborhood groups 24
3.8 PoseFormer architecture demonstration 25
3.9 Demonstration of Strided Transformer framework 26
3.10 Showcase of sequences from Human3.6M dataset 26
3.11 Demonstration of the MixSTE [Zha+22] architecture 27
3.12 Demonstration of the StridedPoseGraphFormer [Ban+23] architecture 28

4.1 Overview of the proposed spatio-temporal graph convolution-based sequence to sequence architecture, TGraphNet 31
4.2 Visulaization of full framework of the proposed TGraphNet and TGraphNet (traj) models 32
4.3 A demonstration of spatio-temporal graphs of pose sequences 33
4.4 Demonstration of the skeleton as a graph with separated neighborhood groups. 35
4.5 Visualization of short-term feature extraction and aggregation in the STGCN block 36

5.1 Visual demonstration of the strided sequence batching strategy 42
5.2 MPJPE in millimeters for the 3D pose of each input frame of TGraphNet 43
5.3 Mean per joint position error under Protocol 1 in millimeters for all joints . . . 47
5.4 Histogram of MPJPE on Human3.6M test set for all actions 47
5.5 Distribution of MPJPE for 4 actions, Sitting, SittingDown, Walking, and Waiting 48
5.6 Model predictions versus ground truth 3D poses for*Walking* and *Sitting* actions 49
5.7 Model predictions versus ground truth 3D poses on MPI-3DHP 51
5.8 Model predictions versus ground truth 3D poses for 2 actions in MPI-3DPW, *Fencing* and *Running for Bus* 52

5.9 Comparison of motion of joints using single-frame and multi-frame output models 54
5.10 Visualization of learned adjacency matrices 55
5.11 Average MPJPE of the Hip joint in millimeters for all actions 58
5.12 Distribution of MPJPE of the global position for 4 actions, Sitting, SittingDown, Walking, and WalkTogether 59
5.13 TGraphNet (traj) predictions versus ground truth 3D poses for *Walking* and *SittingDown* actions 60
5.14 Model predictions versus ground truth 3D poses on MPI-3DHP for TGraphNet (traj) 61
5.15 Model predictions versus ground truth 3D poses of *Fencing* action MPI-3DPW for TGraphNet (traj) 62

A.1 Comparison of scenes from 3DPW and Human3.6M datasets 67

List of Tables

5.1 Comparison of TGraphNet with the state of the art methods 44
5.2 Comparison of TGraphNet with the state of the art using ground truth 2D locations as input 44
5.3 3DPCK and AUC on the MPI-3DHP test set 46
5.4 Comparison of the computational cost of TGraphNet and other state of the art methods 46
5.5 Results of ablation study on architectural parameters 53
5.6 Results of ablation study on different loss functions 53
5.7 Results of ablation study using different input lengths 53
5.8 Comparison of TGraphNet and TGraphNet (traj) in MPJPE and MPJVE on Human3.6M dataset 57
5.9 3DPCK and AUC on MPI-3DHP test set for TGraphNet and TGraphNet (traj) . 58
5.10 Results of ablation study on architectural parameters of TGraphNet (traj) . . . 62
5.11 Results of ablation study on loss functions used to supervise TGraphNet (traj) 62
5.12 Results of ablation study on a different number of input frames to TGraphNet (traj) 63

Chapter 1

Introduction

1.1 Motivation

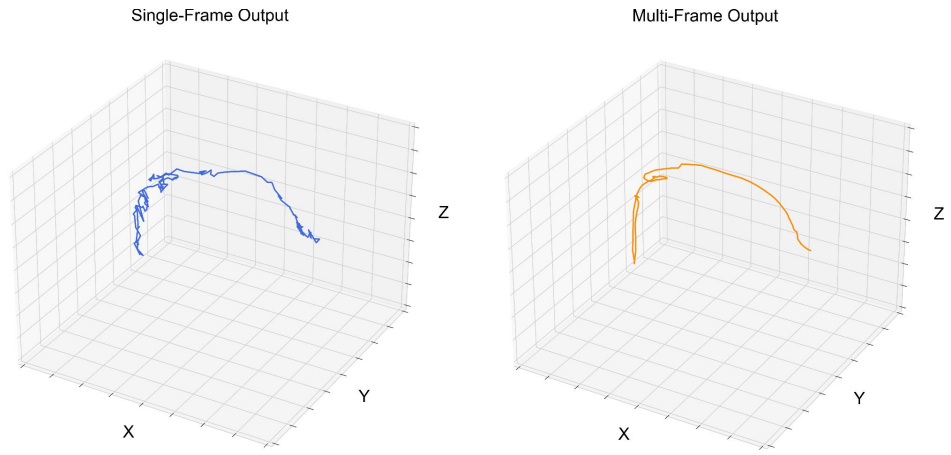

Figure 1.1: 3D Trajectory of the right wrist of the person across time estimated by single-frame and multi-frame output models. The Left shows the trajectory predicted by using single-frame output without temporal continuity enforcement. The right shows the trajectory predicted using multi-frame outputs with supervised smoothness. As it is shown in the plot, the trajectory predicted by the multi-frame output model with enforced continuity is smoother and contains less jitter.

Motion Capture (MoCap) systems have a crucial role in constructing a digital portrayal of the human body and its movement, allowing the examination of intricate motions and the structure of the human body through digital software and tools. The applications of MoCap extend to various fields, such as:

- Sports Analytics: MoCap is used to analyze the body structure and motion of athletes during sports events and training to help improve their performances.
- Media and Entertainment: MoCap allows the incorporation of human actors to create fictional characters in movies and games with realistic motion features.
- Robotics: MoCap is used in robotics to help create more realistic and natural movements for robots.
- Healthcare: MoCap is used in physical therapy and helps in treating various conditions.

Nevertheless, the utilization of MoCap is constrained due to the need for specialized hardware and complex setup. Accurate motion capture often requires expensive marker-based

setups with multiple cameras which ultimately results in high-cost and importable systems. However, recent advancements in Deep Learning (DL), Computer Vision, and 3D Human Pose Estimation (HPE) techniques have made it possible to extend the advantages of MoCap systems to a broader range of users. 3D HPE is the process of recovering a three-dimensional representation of the human body from images or videos. Common representations of a 3D body model range from 3D locations of keypoints to full human body meshes. Typically, 3D HPE is performed using deep learning models that are sophisticated enough to be able to learn an association that ties the 2D information in the image to the 3D representation. The benefits of 3D HPE include cost-effectiveness and the ability to be used with commonly found digital cameras and computers without the need of setting up MoCap systems. 3D HPE techniques can be broadly categorized into direct methods that go from image to 3D pose directly or 2D to 3D lifting methods that first build 2D human pose representation in the image space and then *lift* them to 3D joint locations. 2D to 3D lifting is an inherently ill-posed problem because of the missing depth information in the input 2D joint coordinates. This is called depth ambiguity and can be remedied by restricting the output 3D joint locations to the camera coordinate space. This leads to consistent mapping from input to output joint locations. To build accurate 3D joint locations from the 2D human pose, a method needs to take into account the natural relationships between body parts and joints in the human skeleton. While previous methods exploit the skeletal structure of the human body by using transformers or graph convolutional networks (GCN), they typically process a single frame or an image as input. Some consider multi-frame input but still opt to predict 3D pose corresponding to the central frame [Zhe+21] instead of predicting full 3D motion corresponding to the input. While this leads to promising results, a single-frame output does not allow supervision of the model with motion losses. Therefore, the output motion becomes rough and lacks smoothness (see figure 1.1). Some previous work developed full sequence to sequence 3D HPE systems [Li+23; Zhe+21; Zha+22; Pav+19]. However, they rely on heavy architectures with a large number of parameters that lead to slower performance.

The goal of this thesis is to address the above-mentioned problems by developing a spatial-temporal graph convolution-based 3D HPE architecture named *TGraphNet*. TGraphNet aims to accurately estimate 3D pose by leveraging spatial-temporal relationships in the input data through the application of spatio-temporal graph convolutions. The model incorporates graph convolutions and temporal convolutions within a single framework to enhance structural features extracted from the input 2D pose sequence. Moreover, the model is trained with special loss functions that enforce realistic trajectory of movement of joints. The ultimate goal is to combine lightweight spatial-temporal feature extraction and motion supervision to generate a realistic and continuous sequence of 3D poses from input videos. Additionally, an extension of TGraphNet called *TGraphNet (traj)*, is proposed to predict the global trajectory of the person by estimating the global position of the skeleton from temporal features.

Proposed methods, TGraphNet and TGraphNet (traj) are trained and evaluated on the Human3.6M large-scale 3D HPE datasets and evaluated on multiple benchmark datasets to compare with the current state-of-the-art (SOTA) methods. Additionally, quantitative and qualitative error analysis is performed to highlight the benefits and shortcomings of the methods. Multiple ablation experiments are performed to assess the importance of architectural decisions and their effects on the overall contribution to the performance.

1.2 Outline of the Thesis

The rest of the thesis is structured as follows.

- Section 2 introduces and briefly discusses the formal notations and background necessary to understand the methodology behind the architecture and design choices of the thesis.
- Section 3 discusses relevant related research focusing mainly on 2D to 3D lifting architectures. The work presented in this thesis is compared and contrasted with the related work throughout the section, where necessary.
- Section 4 presents in detail the proposed architecture for the estimation of 3D human pose sequences.
- Section 5 introduces the experimental setup and results for both root relative 3D HPE and global trajectory estimation models.
- Section 6 concludes the results of the work and highlights possible directions for future work

Chapter 2

Foundations and Background

This section aims to introduce the fundamentals that lay at the foundation of the work presented in this thesis. Basic image formation is presented in order to emphasize the relationship between the image and the geometrical object in the camera space. Graph Convolutional Networks and their properties are introduced alongside the foundations of the 3D Human Pose Estimation. The relevance of each topic to 3D HPE is highlighted throughout the section.

2.1 Image Formation

Image formation is the process of mapping a three-dimensional object into the image plane through a sensor. Each location of the physical object corresponds to the location on the 2D image plane. The physical object's surface scatters light, and the light is projected on the image sensor, thus generating a 2D representation of the object on the image plane. The relationship between the object in the scene and the object in the image is essential to 3D HPE. Moreover, the same imaging techniques can be used to acquire a sequence of images or videos for 3D HPE. This section highlights the importance of imaging and what it means in the context of human pose estimation.

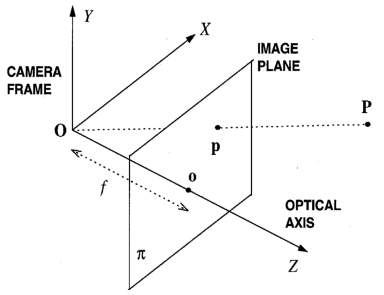

Figure 2.1: A pinhole camera model, adapted from [TV98]. A point **P** is being projected onto the image plane π with principal point **o**.

2.1.1 Perspective Model

A classical model of projection of light onto the sensor is the perspective camera or otherwise called the pinhole camera model [TV98]. It consists of an image plane π and the center of projection **O** which can be thought of as the center of the lens. The distance between the center of the projection and the image plane is called *focal length*. It determines how much of the physical scene is projected onto the lens. The intersection of the optical axis (or the depth axis) and the image plane is called the *principal point* or the image center. These parameters together determine the fundamental equation that describes the projection of the scene onto the lens in such a camera system. The camera frame coordinates of a 3D point $\mathbf{p} = [X, Y, Z]^T$

are then computed as follows:

$$\begin{aligned} x_i &= f\frac{X}{Z} \\ y_i &= f\frac{Y}{Z} \\ z_i &= f \end{aligned} \tag{2.1}$$

Since the depth of the point on the camera frame is always equal to the focal length, one can ignore it and state that the image coordinates of the point are
$\mathbf{p}_i = [x_i, y_i]^T$.

2.1.2 Camera Intrinsic Parameters

The intrinsic parameters of the camera are a set of parameters that characterize the optical, geometric, and digital characteristics of a camera [TV98]. For a perspective camera model, these include:

- Perspective projection defined by the focal length.
- The transformation between the camera image frame coordinates to pixel coordinates governed by the principal point and the pixel height and width.
- The geometric distortion introduced by the lens. These can be neglected if lens distortion is not large or evident.

The relationship between the image pixel coordinates and the camera image plane coordinates can be written in a matrix equation.

$$\begin{aligned} x_{img} &= \frac{x_i}{s_x} + c_x \\ y_{img} &= \frac{y_i}{s_y} + c_y \end{aligned} \tag{2.2}$$

where s_x and s_y are the pixel dimensions, c_x and c_y are the coordinates for the center of the plane. Now, with the full set of equations linking the pixel coordinates to the 3D coordinates of the object available, the importance of those in 2D to 3D lifting HPE can be discussed. In the context of 2D to 3D lifting HPE methods, this relationship between points in the image plane and the physical objects plays a crucial role. Equation 2.1 governs the relationship between 3D joint locations in the camera space and 2D joint locations in the image space. When restricting the problem of 3D HPE to a camera system, the goal is essentially learning to reverse the process of projection of 3D joint locations to the image plane by learning a mapping from pixel coordinates to 3D joint locations. The consistency between the set of joint locations in camera and image spaces makes the 2D to 3D lifting task feasible in a given camera setup.

2.1.3 Sequence of Images

Similar to how a single image is captured using a perspective camera, a stationary camera can capture a sequence of images of physical objects in time. Typically, a sequence of images is captured with a fast capture rate. This is measured in frames per second (fps) which describes how many images are captured in a second. More formally, an image sequence is a sequence of N frames captured at discrete time points $t_k = t_0 + k\Delta t$ where Δt is a fixed time interval

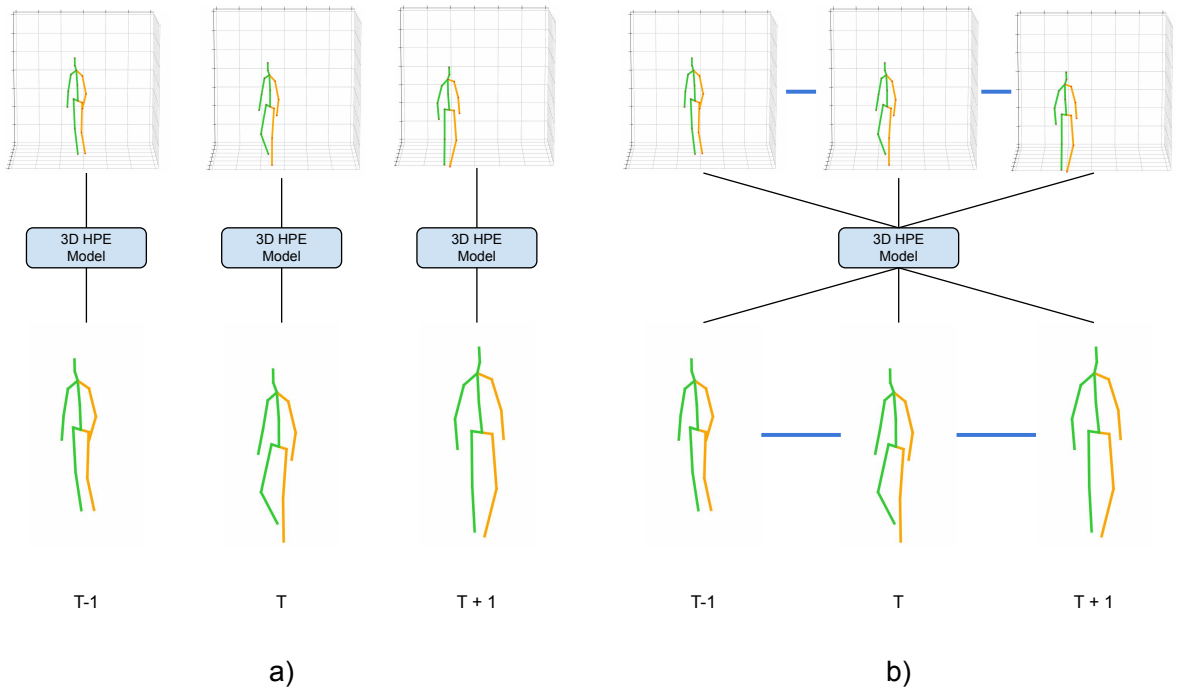

Figure 2.2: An overview of sequence to sequence and single frame 3D HPE pipelines. a) Pose sequence estimation using a single frame 3D HPE model. b) 3D pose estimation using a sequence to sequence model. The black lines represent the input and output of the model while the blue lines demonstrate temporal relationships. In a single frame pipeline (a), a model is run on a sequence one frame at a time ignoring the temporal relationships between input frames and not forcing any relationships between outputs. However, a sequence to sequence (b) model is able to enforce temporal continuity since it processes a sequence entirely.

and $k = 0...N-1$ [TV98]. Small Δt is desired since it allows the capture of the objects' motion more continuously. When the camera is stationary and fixed in the exact location in space, the motion of the objects in the image sequence can provide crucial cues for visual analysis. In the context of 3D HPE, using an image sequence instead of a single frame introduces valuable benefits and challenges. Access to a stream of sequences of images means that one can obtain a sequence of 2D joint locations and process them together instead of individually. One of the benefits is that a model can learn to exploit temporal relationships between sequences of input 2D joint locations to eliminate the effect of noise introduced during 2D joint detection or occlusions. Another crucial benefit is that the motion and continuity of a sequence of 2D joint locations can be used to predict a sequence of corresponding 3D joint locations exhibiting the same temporal relationships and continuity as the input sequence. Figure 2.2 demonstrates this visually. However, processing long sequences of images comes at a price of computational complexity. Efficient neural network architectures are necessary to be able to process long sequences without sacrificing fast runtime.

2.2 Graph Convolutional Neural Networks

Graph Neural Networks (GNN) is a class of Deep Learning architectures that are specifically designed to process graph-structured data. In other words, they can process data structures that contain neighborhood information between entities. The main feature of GNNs is the ability to extract features related to structures found in the graph through message passing between nodes that are connected to each other [KW17; Zha+19a; Wu+21]. GNNs are also widely used in 3D HPE since the human skeleton can be naturally represented as a graph,

with nodes being the joints and edges being the bones that connect them. This representation allows the structural priors of the skeleton to be incorporated into the model. That is, positional features extracted from joints are shared through message passing of GNNs, thus allowing neighboring joints' influence on each other to be captured [ZWT22]. Before the introduction to Graph Convolutional Networks (GCN) defined in [KW17], it is crucial to understand graphs and graph operators such as spectral convolution and spatial convolution. This section acts as an introduction to GNNs, their motivation and foundations, modern GCNs, and their role in 3D HPE.

2.2.1 Motivation of Graph Neural Networks

A graph is a set of nodes connected to each other. It can be formalized as $G = (V, E)$, where V is the set of nodes and E is the set of edges. $e_{ij} \in E, e_{ij} = 1$ means that ith and jth nodes are connected in the graph. A typical way of representing the connectivity of nodes in the graph is through the adjacency matrix $\boldsymbol{A}$. $\boldsymbol{A}_{ij} = 1$ implies that $e_{ij} \in E$. A weighted graph has nonbinary values in the adjacency matrix that represent the weight of the edges. If a graph is undirected, the adjacency matrix is symmetric since edges represent a two-way connection between nodes. However, a directed graph is a graph where edges represent a one-way connection from the source node to the sink node. An undirected graph can be thought of as a directed graph where both directions of an edge connecting two nodes are explicitly specified [Zho+20].

GNNs were inspired by the growing success of convolutional neural networks (CNN), which are able to extract both high and low-level features from grid-like structured data such as images. CNNs employ predefined kernels or filters that extract features from a connected portion in a grid [Wu+21]. The constraint of CNNs is that the input data has to be structured and regular. The forward pass of a CNN also assumes that processed parts of the grid are fully connected. It can be considered a weighted combination of entries in the grid spanned by the convolutional filter.

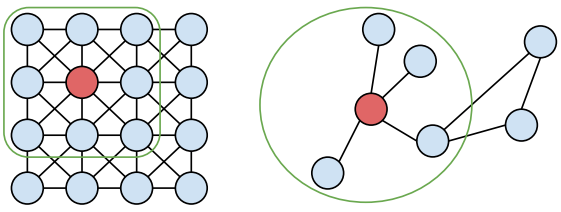

Figure 2.3: A 2D convolution versus a GNN. In a), a convolutional filter of dimension $3x3$ is demonstrated. It operates on a pre-defined part of a grid and computes a weighted combination of elements in the patch highlighted by the green rectangle. b) shows a GNN being applied to extract features for the red node using its neighborhood. Pre-defined kernel approach used in a) is infeasible for irregular graphs as a node may have a varying number of neighbors.

However, the ability to process irregular structures necessitates the introduction of specialized architectures. This is necessary since graphs, unlike grids, are invariant to permutation, have irregular connectivity, and may have different numbers of nodes [KW17; Bru+14]. First, a formal introduction to the properties and key graph-related operations is necessary. These operations are the foundation for two major types of graph neural networks.

2.2.2 Graph Operations

Spectral Graph Convolution: A crucial definition for undirected graphs is the Laplacian matrix $\boldsymbol{L}$. $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{D}$ is the diagonal degree matrix representing the number of edges connected to the node, $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$. The symmetrically normalized graph Laplacian is defined as $\boldsymbol{L}_{norm} = \mathbf{Id}_n - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ Since $\boldsymbol{L}_{norm}$ is symmetric and positive semidefinite, it has N real eigenvectors and eigenvalues. Thus, it can be factored as $\boldsymbol{L}_{norm} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$. Here,

$\mathbf{\Lambda}$ is the diagonal matrix of the eigenvalues of the Laplacian matrix and $\boldsymbol{U}$ is the matrix with columns as the eigenvectors. Laplacian matrix is used in GNNs with a focus on spectral graph convolutions which process graph features in the spectral domain. [Wu+21].

Graph signal is a vector $\boldsymbol{x} \in R^n$ which contains features for all n nodes in the graph. The graph Fourier transform of the graph signal $\boldsymbol{x}$ is then defined as $\mathcal{F}(\boldsymbol{x}) = \hat{\boldsymbol{x}} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}$. The inverse of $\mathcal{F}$ is $\mathcal{F}^{-1}(\hat{\boldsymbol{x}}) = \boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}$ The graph Fourier transform projects the input graph signal to the orthonormal space, the basis of which is formed by eigenvectors of the normalized graph Laplacian. Then, a graph convolution in the spectral domain is defined as:

$$\begin{aligned} \boldsymbol{x} *_G \boldsymbol{g} &= \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{x}) \odot \mathcal{F}(\boldsymbol{g})) \\ &= \boldsymbol{U}(\boldsymbol{U}^{\mathrm{T}} g \odot \boldsymbol{U}^{\mathrm{T}} x) \end{aligned} \tag{2.3}$$

where $(*_G)$ denotes the convolution on the graph G. Equation 2.3 can be further simplified by defining a filter $\boldsymbol{g}_\theta = diag(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x})$. Thus, the equation can be rewritten as follows

$$\boldsymbol{x} *_G \boldsymbol{g} = \boldsymbol{U}\boldsymbol{g}_\theta\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} \tag{2.4}$$

Equation 2.4 lays at the foundation of *Spectral-Based* graph convolutional networks [Bru+14]. Spectral-Based graph convolutional networks use the same equation but differ only in the choice of the filter $\boldsymbol{g}_\theta$ leading to various kinds of architectures [Bru+14; ZWT22].

Spatial Graph Convolution: Analogous to spectral-based convolution described in the previous section, the spatial-based method defines a convolution based on the neighborhood of nodes in the graph [Zha+19a]. It is simple to describe, and intuitively, it aggregates features of neighbors of a node.

$$\hat{\boldsymbol{x}}_i = \boldsymbol{w}_{i,i}\boldsymbol{x}_i + \sum_{j \in N(i,k)} \boldsymbol{w}_{i,j}\boldsymbol{x}_j \tag{2.5}$$

where $N(i,k)$ is the k hop neighborhood of the ith node in the graph. In other words, $N(i,k)$ is the set of neighbor nodes with a distance less than or equal to k. $w_{i,j}$ are the weights assigned to each neighboring node and the central node itself.

2.2.3 Types of Graph Convolutional Networks

Spectral-Based Graph Convolutional Networks: The Spectral Network proposed in [Bru+14] uses a learnable diagonal matrix as parameters for $\boldsymbol{g}_\theta$ in 2.4. This leads to several problems. The equation 2.4 is costly to compute since the spectral decomposition of $\boldsymbol{L}_{norm}$ is of complexity $O(n^3)$. Also, learned filters are not spatially localized in this implementation. Moreover, this method strictly depends on the structure of the graph. Any perturbation of the graph changes the eigendecomposition, hence requiring a different filter [Wu+21].
ChebNet [DBV16] proposes to approximate the filter $\boldsymbol{g}_\theta$ by using Chebyshev polynomials of the diagonal matrix of eigenvalues $\mathbf{\Lambda}$ of the normalized Laplacian.

$$\boldsymbol{x} *_G \boldsymbol{g} = \boldsymbol{U}\left(\sum_k \theta_k T_k(\tilde{\mathbf{\Lambda}})\right)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} \tag{2.6}$$

Here, $\tilde{\mathbf{\Lambda}} = 2\mathbf{\Lambda}/\lambda_{max} - \mathbf{Id}_n$. If we note $\tilde{\boldsymbol{L}} = 2\boldsymbol{L}_{norm}/2\lambda_{max} - \mathbf{Id}_n$, it can be proved that $T_k(\tilde{\boldsymbol{L}}) = \boldsymbol{U}T_k(\tilde{\mathbf{\Lambda}})\boldsymbol{U}^{\mathrm{T}}$. Then, the ChebNet equation simplifies further to

$$\boldsymbol{x} *_G \boldsymbol{g} = \sum_{k=0}^{K} \theta_k T_k(\tilde{\boldsymbol{L}})\boldsymbol{x} \tag{2.7}$$

ChebNet filters, in contrast to the spectral network proposed in [Bru+14] are localized in space, meaning that ChebNet extracts local features regardless of the graph size [Wu+21].

Spatial-Based Graph Neural Networks: As mentioned in section 2.2.2, spatial-based methods largely follow the same ideology as spectral-based methods and consider neighbors of nodes for feature extraction. However, there is a striking difference between the two approaches. In spatial-based methods, feature extraction and message passing happen directly in vertex rather than spectral space. The convolution operation depicted in figure 2.3 is a simple example of a spatial-based convolutional network. The feature of the central node in a $3x3$ grid is computed using its spatial neighbors. A general spatial-based graph neural network example is the Message Passing Neural Network proposed in [Gil+17]. It uses the neighborhood information of the graph to define the message passing scheme for nodes. K steps of message passing steps are run to enable long-distance information aggregation. The message passing rule is defined as

$$\boldsymbol{h}_v^{(k)} = U_k(\boldsymbol{h}_v^{(k-1)}, \sum_{v \in N(v)} M_k(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{h}_u^{(k-1)}, \boldsymbol{e}_{\boldsymbol{vu}})) \tag{2.8}$$

where $\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v$ and U_k and M_k are neural networks with learnable parameters, such as Multilayer Perceptrons (MLP). $\boldsymbol{e}_{\boldsymbol{vu}}$ denotes the edge features connecting nodes v and u [Gil+17].

The equation 2.8 is general and is open to the freedom of adjusting the learnable functions [Duv+15; Kea+16; Gil+17]. One benefit of the spatial-based approaches is the invariance of models to the size of the input graph and the number of neighbors of nodes. If the input graphs to the network vary in size and structure, spatial-based methods can be readily used, whereas spectral-based methods rely on the Laplacian matrix, meaning they require graphs with the same structure and size as input.

Graph Convolutional Network (GCN) proposed by Kipf et al. in [KW17] bridges the gap between spectral-based and spatial-based methods. It can be interpreted through both paradigms due to its algorithm. In [KW17], Kipf et al. propose to simplify the definition given in 2.7. They truncate the sum to only use first order polynomials, ($K = 1$). They further assume that $\lambda_{max} = 2$ since the network can adapt to this assumption during training. Equation 2.7 simplifies to

$$\boldsymbol{x} *_G \boldsymbol{g} \approx \theta_0 \boldsymbol{x} + \theta_1(\boldsymbol{L}_{norm} - \mathbf{Id}_n)\boldsymbol{x} = \theta_0 \boldsymbol{x} - \theta_1 \left(\boldsymbol{D}^{(-\frac{1}{2})}\boldsymbol{A}\boldsymbol{D}^{(-\frac{1}{2})}\right)\boldsymbol{x} \tag{2.9}$$

Further, θ_0 is assumed to be equal to $-\theta_1$, which restrains the number of parameters and prevents overfitting in practice [KW17]. The simplified form of the equation is as follows.

$$\boldsymbol{x} *_G \boldsymbol{g} \approx \theta \left(\mathbf{Id}_n + \boldsymbol{D}^{(-\frac{1}{2})}\boldsymbol{A}\boldsymbol{D}^{(-\frac{1}{2})}\right)\boldsymbol{x} \tag{2.10}$$

Kipf et al. found out that using $\mathbf{Id}_n + \boldsymbol{D}^{(-\frac{1}{2})}\boldsymbol{A}\boldsymbol{D}^{(-\frac{1}{2})}$ causes instability issues while training. To circumvent this issue they employ a *renormalization trick*: $\mathbf{Id}_n + \boldsymbol{D}^{(-\frac{1}{2})}\boldsymbol{A}\boldsymbol{D}^{(-\frac{1}{2})} \longrightarrow \tilde{\boldsymbol{D}}^{(-\frac{1}{2})}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{(-\frac{1}{2})}$ with $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \mathbf{Id}_n$ and $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$

Finally, setting $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{(-\frac{1}{2})}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{(-\frac{1}{2})}$ the forward pass of GCN can be stated as

$$\boldsymbol{H} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}) \tag{2.11}$$

where $\boldsymbol{H} \in R^{n \times h}$ is the hidden state of nodes in a graph after a forward pass, $\boldsymbol{X} \in R^{n \times \mathrm{f}}$ is the input graph signal where each of n nodes has f dimensional features and $\boldsymbol{W} \in R^{\mathrm{f} \times h}$ is the learnable weight matrix. The result is passed through a non linearity σ, which can

be arbitrarily chosen. A common choice is the *ReLU* function. Stacking several such layers enables the network to perform long-distance message passing, much like MPNN discussed above. For a better perspective, one can split the equation above into two crucial parts.

$$\hat{\boldsymbol{H}} = \boldsymbol{X}\boldsymbol{W} \tag{2.12}$$

$$\boldsymbol{H} = \sigma(\hat{\boldsymbol{A}}\hat{\boldsymbol{H}}) \tag{2.13}$$

The first part of the equation can be described as feature extraction, where the weight matrix is multiplied with node attributes to generate new features for nodes. The second part is the message passing part, where multiplication with the adjacency matrix $\hat{\boldsymbol{A}}$ combines features of neighboring nodes allowing nodes to influence each other's features.
To see why this definition can be viewed as a spatial-based method, it can be rewritten as follows

$$\boldsymbol{h}_v = \sigma\left(\sum_{u \in \{N(v) \cup v\}} \hat{\boldsymbol{A}}_{v,u} \boldsymbol{W}^{\mathrm{T}} \boldsymbol{x}_u\right) \tag{2.14}$$

From the spatial-based perspective, GCN can be considered an aggregation of features from the node's neighborhood [Wu+21]. It can be noted, however, that GCN uses an adjacency matrix multiplication in the equation. Hence, unlike the more general approach defined in equation 2.8, it is only suitable if the input graph structure is consistent across different inputs. In the context of 3D HPE, the human skeleton has a consistent topology and typically includes 17 joints, meaning that it can be modeled using a 17×17 adjacency matrix. Hence, GCNs can be used to process human skeletons as graphs.

2.2.4 Extensions to Graphs and Graph Convolutional Networks

Spatio-Temporal Graph: Spatio-Temporal graphs are an extension of regular graphs. They are useful when modeling relationships of entities across time is necessary. Spatio-temporal graphs are attributed graphs, meaning that nodes possess attributes that evolve over time [Wu+21]. By representing sequential data as spatio-temporal graphs, earlier mentioned graph processing architectures can be applied to such data. This approach allows the modeling of both short-term and long-term temporal relationships using graph neural networks. When it comes to 3D HPE, spatio-temporal graphs allow the processing of a sequence of skeletal structures extracted from videos. In doing so, input to graph neural networks becomes enriched with temporal data which results in increased performance of 3D HPE models [Cai+19a].

Formally, a spatio-temporal graph can be defined as an attributed graph $G = (V, E, X)$ where $V = \{v_{ti} \mid t = 1, \dots T; i = 1, \dots, N\}$ denotes set of nodes across time. Specifically, $v_{t_1 i}$ and $v_{t_2 i}$ the same node at time points t_1 and t_2. E is the set of edges connecting nodes across time and space. $\boldsymbol{X} \in R^{(TN) \times \mathrm{f}}$ is the attribute matrix containing attributes or features of the nodes. Spatio-temporal graphs can be modeled with adjacency matrices as well. In line with the aforementioned definition, the matrix $\boldsymbol{A} = (a_{ij})_{T \times N, T \times N}$ is a matrix of dimensions $(T \times N, T \times N)$ that represents the connectivity or links between the spatio-temporal nodes. With this definition, $\boldsymbol{A}$ and $\boldsymbol{X}$ can be plugged into the GCN forward pass equation 2.11 and obtain a spatio-temporal GCN (*STGCN*) [Cai+19a]. However, suppose the input sequence is significantly long. In that case, it becomes infeasible to represent the whole sequence as a single spatio-temporal graph since the adjacency matrix grows $\mathcal{O}(T^2)$ with respect to the temporal dimension T. This leads to large memory and resource consumption when carrying

out the adjacency matrix multiplication step. The sequence can be split into shorter spatio-temporal graphs to circumvent this issue. Then, short-term spatial-temporal features can be extracted using STGCN on individual parts. Later, these short-term features can be aggregated to form more global, long-term spatial-temporal features. This leads to local-to-global temporal feature extraction where first adjacent temporal neighbors are processed, and then the extracted local features are combined across time.

Weight Unsharing in GCNs: One limitation of vanilla GCNs characterised by equations 2.11 is that the weight matrix $\boldsymbol{W}$ is shared for all nodes in the graph. This means that feature transformation is performed in the same manner for all nodes regardless of their semantics. In the context of 3D HPE, this means that legs and arms, for instance, share the same feature transformation. This approach can be problematic as joints and their interactions are complex and single shared transformation is limiting[Liu+20]. On the other hand, weight sharing can prevent overfitting because of less complex feature extraction capabilities. Also, having single a weight makes architectures more lightweight with a low number of parameters.
To solve the limitations of weight sharing in GCNs, one can train a weight matrix for all possible pairs in the graph and achieve completely unshared weights [Liu+20]. Thus, one can arrive at the following a feature transformation and message passing rules in the graph

$$\boldsymbol{h}_v = \sigma\left(\sum_{u\in\{N(v)\cup v\}} \hat{\boldsymbol{A}}_{v,u}\boldsymbol{W}_{v,u}^{\mathrm{T}}\boldsymbol{x}_u\right) \tag{2.15}$$

Where $\boldsymbol{W}_{v,u}$ is the linear feature transformation specific to interaction between nodes v and u. This can lead to more powerful feature extraction but the number of parameters grows significantly. Full weight sharing and complete weight unsharing are the two extreme variants of feature transformations in GCNs. While full weight sharing hinders performance as complex interactions between joints are not captured, complete weight unsharing leads to overly complicated feature extraction. Thus, a middle ground for weight unsharing involves introduction of weights specific to certain body parts as opposed to each pair of individual joints. That is, joints can be grouped into categories based on their semantics and weights can be shared within each category. For example, legs and arms can be separated into two groups with specific weights. The following equation captures this idea

$$\boldsymbol{h}_v = \sigma\left(\sum_{u\in\{N(v)\cup v\}} \hat{\boldsymbol{A}}_{v,u}\boldsymbol{W}_{g(v,u)}^{\mathrm{T}}\boldsymbol{x}_u\right) \tag{2.16}$$

Where $g(v,u)$ corresponds to the group where joints v and u belong. So, $\boldsymbol{W}_{g(v,u)}$ is the feature transformation specific to the group $g(v,u)$.

Learnable and Split Adjacency Matrix: Another limitation of the vanilla GCN is that the adjacency matrix is fixed and relationships between nodes are predetermined. However, in some applications, making the adjacency matrix dynamic and learnable during training can lead the GCN to discover useful relationships between nodes itself. For instance, some joints in the human skeleton such as left and right shoulders, while not connected physically are still related to each other via symmetries [BGK21]. Allowing the adjacency matrix in equation 2.11 be a trainable weight matrix can increase the performance of the model depending on the task.

Additionally, like unshared weights, the adjacency matrix can also be split into groups alongside the feature transformation matrix to allow both non uniform feature extraction and message passing [Cai+19a; XT21; BGK21]. The following equation formalizes a non uniform graph convolution combining both adjacency matrix splitting and weight unsharing

$$\boldsymbol{H} = \sigma\left(\sum_{g \in G} \hat{\boldsymbol{A}}_g \boldsymbol{X} \boldsymbol{W}_g\right) \tag{2.17}$$

Where G represents the set of groups the graph has been partitioned into. Thus, $\boldsymbol{W}_g$ and $\boldsymbol{A}_g$ represent group specific weights and adjacency matrices. Each group g has a specific feature transformation and interactions due to the splitting of weights and adjacency matrices. This method is a more efficient alternative to complete weight unsharing as there are only g weight matrices to be learned instead of a matrix for each pair of nodes.

2.3 Transformers

Since Transformers are very popular in contemporary research in 3D HPE, a brief introduction and key concepts are presented in this section. Moreover, Transformers are closely related to graph neural networks since they also model relationships between entities represented as feature vectors. Precisely this ability makes transformers one of the most popular architectures used in 3D HPE [Zha+22; Zhe+21; Li+23].

Self Attention is the core of the transformer architecture [Vas+17]. It is an operation that is run on a set or a sequence of features to model interactions between all entities. Self attention allows each entity to capture features from others through a global interaction mechanism. For all entities, also called tokens, three feature vectors are computed.

$$\boldsymbol{Q} = \boldsymbol{W}_q \boldsymbol{X} \qquad \boldsymbol{K} = \boldsymbol{W}_k \boldsymbol{X} \qquad V = \boldsymbol{W}_v \boldsymbol{X}$$

Here $\boldsymbol{W}_k, \boldsymbol{W}_q, \boldsymbol{W}_v$ are learnable weight matrices allowing the network to dynamically readjust interaction between tokens in the input. $\boldsymbol{Q}$ is the query matrix, $\boldsymbol{K}$ is the key matrix and $\boldsymbol{V}$ is the value matrix. The degree of interaction between each pair of tokens is governed by the similarity of their respective *key* and *query* vectors in $\boldsymbol{Q}$ and $\boldsymbol{K}$. So, interaction between pair of entities x_i and x_j is proportional to $\boldsymbol{Q}_i^{\mathrm{T}} \boldsymbol{K}_i$. These pairwise interactions are collected into a matrix, often called attention matrix, containing pairwise interactions between tokens. This matrix is much like the adjacency matrix for graphs with one major difference. The adjacency matrix is often pre-defined and does not depend on the nodes in the graph while self-attention learns pairwise interaction conditioned on the input tokens. This makes the transformer more flexible but also computationally expensive as opposed to GCNs.
The final computation formula for self-attention is the following:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{SoftMax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{2.18}$$

Which aggregates value features V for all tokens according to the learned interconnection. In the context of 3D HPE, either joints are treated as tokens [Zha+22] or the whole input skeleton is embedded into a token [Zhe+21] for spatial and temporal feature extraction.

2.4 Human Pose Estimation

Human pose estimation has been gaining popularity as a research field in recent years with the evolution of deep learning methods that outperform classical computer vision methods.

Human pose estimation is the task of obtaining the human body posture from given sensor data, typically videos and images. It provides valuable geometric and motion information widely used in fields such as augmented reality, sports analytics, and healthcare, among others [Wan+21].

Even though human pose estimation models have been showing promising results recently, there are still research gaps to be filled for further improvement. This section introduces crucial concepts in human pose estimation, its variants, and challenges that still remain to be solved today. The section is mainly focused on 3D HPE, which is the focus of the research.

2.4.1 Human Body Models

The first step towards human pose estimation is the modeling of the body. Human bodies are complex since they are non-rigid, have large degrees of freedom for movement, and vary in appearance and shape [CTH20]. Thus, human body models need to be able to capture the pose, shape, and movement of human bodies. Human body models range from sparse representations, which include several key joints connected to each other, to arbitrarily complex models that represent both shape and posture in a dense manner, such as meshes. Each has its own advantages and drawbacks. The advantage of sparse models is the availability and ease of modeling, while dense models capture richer information and are harder to utilize. A good model does not necessarily need to be complex and model all aspects of the human body since some tasks, such as action recognition and digital therapy supervision, can be solved with sparse representations [CTH20]. Most commonly used human body models in the literature are as follows:

- **Kinematic Model**: Kinematic model, otherwise known as the skeleton model, is a sparse representation typically including a set of joints (between 10 to 30) following the body's skeletal structure. This representation may optionally include bone orientation or rotations. This type of representation is widely used both in 2D and 3D HPE, and most 2D to 3D lifting models use it. Multiple large-scale datasets incorporate this representation [CTH20; Zhe+22; Mar+18; Meh+17; Ion+14].
- **Contour Based Model**: This type of modeling was prevalent in earlier work and used geometric shapes such as rectangles were used to approximate boundaries of human bodies [CTH20].
- **Volumetric Models**: Modern volumetric models use dense meshes that serve as an accurate representation of both appearance and posture. Since meshes are complex and high dimensional objects, several statistical models have been successfully developed to embed them using a small set of parameters. One such model is the Skinned Multi-Person Linear model (SMPL) [Lop+15], which uses a set of shape parameters that determine the appearance of the mesh and a set of parent relative joint rotation vectors that govern the posture.

2.4.2 Categories of 3D Human Pose Estimation

3D HPE methods can be categorized into several groups depending on the types of input, output, and task that they perform. 3D HPE methods can be either multi-person or single-person. Multi-person pose estimation aims to reproduce the posture of multiple persons

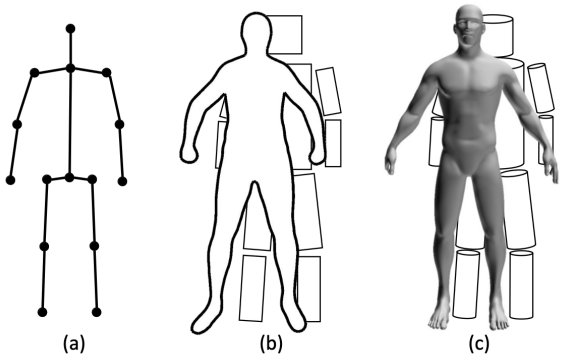

Figure 2.4: Popular human body models used in human pose estimation. a) kinematic model, b) contour based models, c) Volumetric models. Adapted from [CTH20]

from the media at the same time. Unlike single-person methods, a crucial challenge is an association of detected body parts or joint locations to correct persons in the input [CTH20]. Methods can further be divided according to the types of used models. Methods are either two-stage or single-stage. Single-stage methods directly estimate human pose from the input source without intermediate stages. In contrast, multi-stage models move from input to an intermediate representation, then use it to estimate the final 3D human pose. 2D to 3D lifting approaches belong to the second category since the pipeline involves first obtaining a 2D pose from the source, then estimating a 3D pose from 2D joint locations.

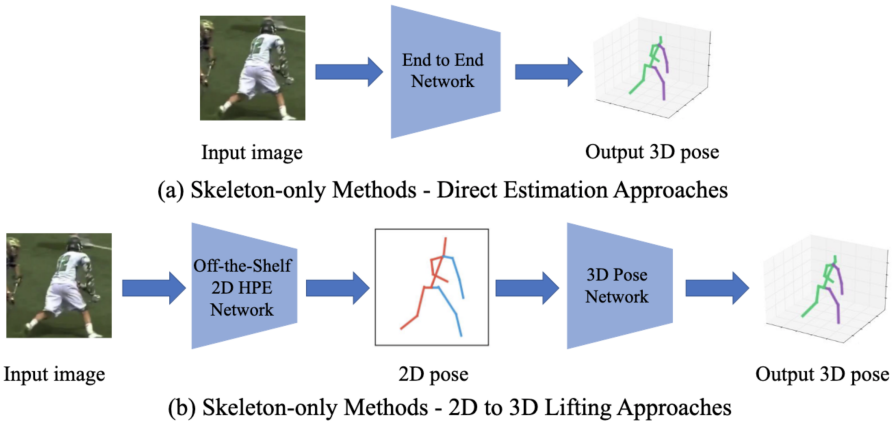

Figure 2.5: Types of 3D HPE models. a) shows models that use the input to directly estimate 3D human pose. b) represents models that use an off the shelf 2D pose estimator to obtain 2D skeletons and then estimate 3D posture from those. Adapted from [Zhe+22]

An important category of 3D HPE methods are *Multi-Frame* methods which are essential to this thesis. In contrast to single-frame models, multi-frame models work with sequences of images or videos. They have the advantage of using temporal relationships between inputs leading to fewer prediction errors. Moreover, the estimation of a sequence of 3D poses allows models to enforce temporal continuity and consistency across frames leading to smoother motion. Figure 2.2 demonstrates differences between single frame and multi frame models.

2.4.3 Challenges in 3D Human Pose Estimation

Despite recent impressive progress in the field of 3D HPE, significant challenges still remain to be solved. These challenges range from accurate data collection with sample variety to the development of neural models that are able to perform well and remain computationally efficient at the same time. These challenges are discussed below. A separate accent is made on the challenges and limitations regarding efficient 2D to 3D lifting methods.

- **Data Collection**: Neural models require an abundance of accurate annotated data for training. 3D HPE models, in particular, necessitate the creation of datasets that contain accurate 3D representations of human bodies, be they joint locations or meshes. Thus, specialized settings and hardware for motion capture are used to capture a variety of human poses. This complicates data collection, and as a result, human poses are captured in restricted settings, usually indoors [Ion+14; CTH20]. Hence, the variety of captured data is limited, leading to a generalization gap in models where models perform poorly on the task when it comes to estimating poses in the wild. Accurate capture of in-the-wild poses for evaluation and training of models has the potential of increasing the performance of 3D HPE models [Mar+18].

- **Computational Efficiency**: Real-time applications depending on 3D HPE tasks, such as visual tracking, require real-time performance. 2D to 3D lifting methods are dependent on accurate 2D pose estimators, where improvements in computational efficiency remain necessary. Moreover, 2D to 3D lifting methods themselves should tackle the challenge of arriving at more efficient architectures having fewer parameters [Zhe+22].

- **Lack of Smoothness**: Visual tracking applications require smooth sequence of 3D poses. However, the majority of 2D to 3D lifting methods utilize no temporal contexts in the input leading to less continuous outputs. Evaluation metrics such as commonly used mean per joint position error (MPJPE) do not capture the smoothness of the output pose sequence. Thus, the facilitation of evaluation schemes that also measure the realism of the output 3D poses as a sequence is necessary to overcome this challenge [Zhe+22].

- **Depth Ambiguity**: This issue concerns 2D to 3D lifting methods. Depth ambiguity refers to the problem of loss of depth cues when projecting 3D data to the image space. As a consequence, multiple 3D skeletons can correspond to the 2D input. To circumvent this issue, 2D to 3D lifting methods are constrained to predict the output in the camera space. Moreover, supervision with reprojection loss that ensures consistency between predicted output and input poses also aids. This restriction, however, has a drawback causing a generalization gap. Constraining output and input to certain camera setups hinder the performance of models if the camera setup of captured frames differs significantly from the one used during training. This issue is demonstrated and discussed in later sections of the work.

- **Ignoring Global Position of Root Joint**: 2D to 3D lifting methods mostly focus on obtaining relative root coordinates of joints of the skeleton. The root joint is typically selected to be the hip. The position of the said root joint, however, remains ignored in the 3D representation. However, when the application requires tracking of the trajectory in the 3D space, this is not feasible as the global position of the skeleton is also needed. 3D coordinates of the root joint are harder to estimate because of depth ambiguity and lack of cues, such as symmetries and bone length in the input. This thesis explores a potential solution to this challenge. Usage of the temporal context of the

input that contains information on the trajectory of the skeleton in the image space and ensuring consistency with a projection loss between the predicted 3D skeleton and input 2D as a potential solution is explored in the later sections.

Chapter 3

Related Work

This section provides a coverage of recent methods used in 3D HPE with specific focus on 2D to 3D lifting methods. Both single-frame and multi-frame based models are discussed with their limitations and strengths.

3.1 Single Frame Models

3.1.1 GCN Based Methods

2D to 3D lifting methods started gaining popularity after Martinez et al. showed in [Mar+17] that restricting the target joint locations to the camera space makes it possible to train a network that solves 3D HPE tasks with reasonable accuracy. Specifically, they proposed a simple multi-layer perceptron to lift 2D joint locations from a single monocular image, and even such a simple architecture demonstrated impressive results. Many GCN-based methods follow the same principle but replace the MLP with more sophisticated architectures to represent and utilize complex relationships between joints in the input.

Semantic Graph Convolutional Networks (SemGCN) [Zha+19b]. is one of the seminal works that led to adoption of GCNs in 3D HPE. The authors of SemGCN circumvent the issue of full weight sharing discussed in section 2.2.4 by introducing a set of learnable multiplicative weight vectors per each joint. Figure 3.1 visually demonstrates the modified GCN operation with learnable weights $\boldsymbol{a}_i$. Specifically, for each node v_i in the input skeletal graph, SemGCN learns a multiplicative weight vector $a_i \in R^{\mathrm{h}}$. Here, h is the hidden dimension of features learned in the network. Vectors $\boldsymbol{a}_i$ are then multiplied with hidden features of joints in an elementwise manner to compute joint specific features before shared linear transformation, $\hat{\boldsymbol{h}}_{v_i} = \boldsymbol{a_i} \odot \boldsymbol{h}_{v_i}$, $\boldsymbol{h}_{v_i} = \boldsymbol{W}^{\mathrm{T}}\hat{\boldsymbol{h}}_{v_i}$. SemGCN addresses the limitation of fully shared transformation, however, it is still restricted by a fixed adjacency matrix which

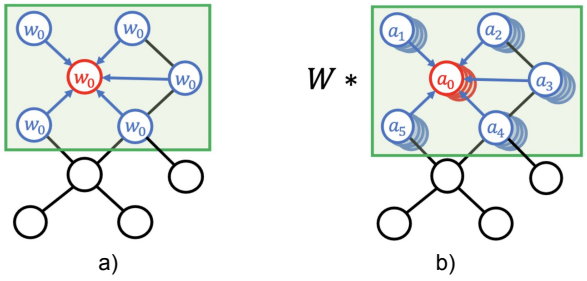

Figure 3.1: Demonstration of Graph Convolution operation used in Semantic Graph Convolutional Networks [Zha+19b]. Typical GCN operation is depicted in a) where weighted features of nodes are aggregated in a neighborhood. SemGCN proposes to learn a channel wise multiplicative weight $\boldsymbol{a}_i$ for each node before applying a shared feature transformation $\boldsymbol{W}$. This method allows the network to extract single node-level features unlike conventional GCN feature transformation is shared for all nodes.

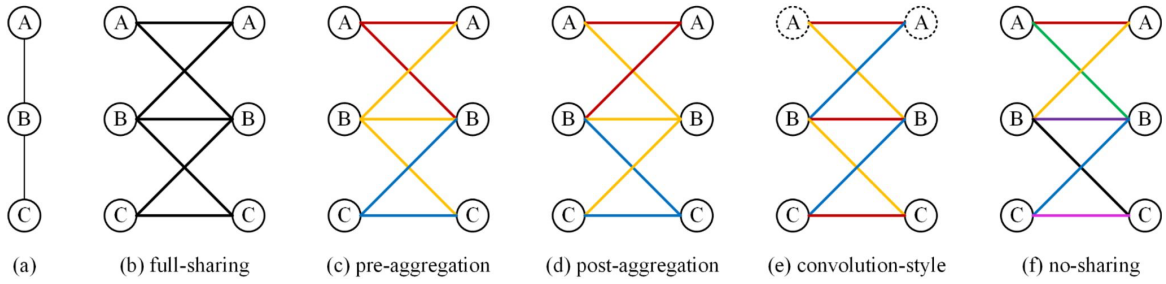

Figure 3.2: Visual demonstration of various weight sharing in GCNs explored by Liu et al. in [Liu+20]. a) demonstrates a simple 3-node graph. b) shows the full sharing scheme where all the weights depicted in black lines are shared between all nodes. c) and d) correspond to pre-aggregation and post-aggregation respectively. Either input or output features are shared and colored lines demonstrate the relationship between weights and nodes. e) shows the convolutional style where nodes share weights depending on their displacement. In f), no weights are shared. Adapted from [Liu+20].

prevents the network from learning relationships between joints that are not directly established by the skeletal structure. Moreover, multiplicative weights per node are too specific and ignore relationships between groups of joints such as arms and legs. These limitations are addressed in the following works.

Pre-Aggregation Graph Convolution by Liu et al. [Liu+20] highlights and analyzes the limitations of weight sharing in GCNs like [KW17]. The authors propose several methods that decouple weights in the traditional GCN formula and contrast them with fully shared weights. They are listed below. Figure 3.2 offers a visual demonstration of these strategies.

- **Pre-Aggregation**: Pre-Aggregation strategy first transforms the input features of each joint depending on the joint type. Then, features of neighbor joints are aggregated after transformation, hence the name.
- **Post-Aggregation**: Post-Aggregation, in contrast to pre-aggregation, first aggregates features of neighboring nodes, then applies a joint-specific feature transformation.
- **No-Sharing**: This strategy eliminates weight sharing completely by introducing separate weight matrices for each pair of neighboring joints. This strategy is the combination of the two above, and it has the most amount of learnable parameters.
- **Convolution Style**: This strategy uses weights specific to displacement between nodes i and j in the graph by defining so called displacement value $d(i,j)$ between two nodes. This formulation needs coordinate values for each node to be able to calculate displacement values. Thus, they define the coordinate of each joint as the length of the shortest path from the said joint to the root of the skeleton, the pelvis. So, joints of arms and legs have larger coordinates, while joints closer to the pelvis have lower coordinates. $d(i,j) \in \{-1,0,1\}$, which intuitively separates the relationship between joint and its neighbors into 3 categories, namely: self-connection if $d(i,j) = 0$, connection to the joint closer to the root, $d(i,j) = 1$ and connection to the joint further away from the joint, $d(i,j) = -1$.

The authors observed that the pre-aggregation technique shows the best performance. Despite tackling the issue of limiting shared weights, the authors do not explore the idea of splitting the skeletal graph into multiple parts via decoupled adjacency matrices as discussed in 2.2.4. Much like SemGCN, the relationships between nodes are bound to the static adja-

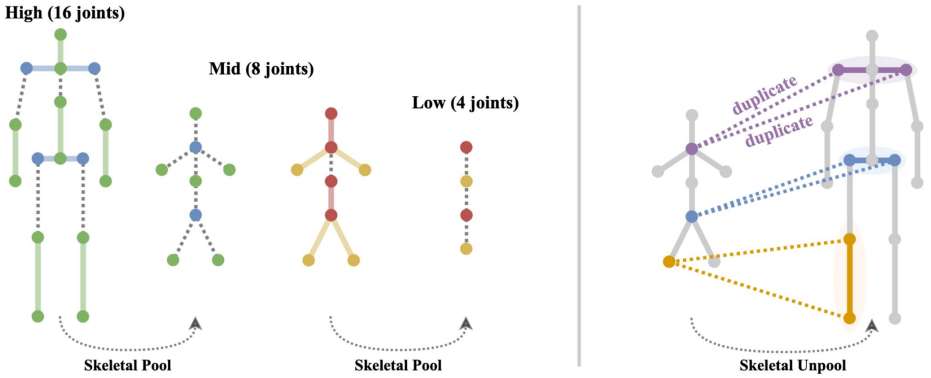

Figure 3.3: Illustration of skeletal pooling and unpooling operations. The left part demonstrates the pooling operation which aggregates the skeleton into a broader structure enabling global feature extraction. Joints of the same color are pooled using a max pooling operator. Two pooling operations are carried to transit from fine-grained to coarse-grained structure allowing multi-scale feature extraction. The right part shows the reverse of the pooling operation. The coarse grain structure is unpooled back to the original fine grain structure. Pooling and unpooling layers are stacked onto each other to achieve interplay between local and global features. Adapted from [XT21]

cency defined according to the skeletal structure.

PoseGraphNet [BGK21], in contrast to the methods discussed above, explores the idea of non-uniform graph convolutions and learnable adjacency matrices by splitting the graph into several groups. Each group has a specific learnable transformation matrix and an adaptive adjacency matrix defining the neighborhood. The authors separate the connections into: a) self links, where joints are only connected to themselves, b) parent links, where each joint is connected to its parent joint (joint closer to the pelvis), c) child links, where each joint is connected to its child joint (joint further away from the pelvis). This scheme is much like the *convolution-style* weight sharing strategy in [Liu+20] discussed above. Learnable adjacency matrices per neighborhood allowed the network to establish relationships between joints that are not present in the skeletal structure. For example, right and left feet became connected in the adjacency matrix during the learning process, whereas they are not physically connected in a human body [BGK21]. This allows the model to discover and utilize spatial relationships between joints in the skeleton to increase the performance of the model.

Graph Stacked Hourglass Network by Xu et al. [XT21] highlights the importance of hierarchical feature extraction in the model for 3D HPE. Unlike other GCN based 2D to 3D lifting methods, graph stacked hourglass operates on three levels of skeletal scale by utilizing pooling and unpooling operations introduced in the paper. This allows the model to learn features that are both local to small neighborhoods of joints and more global features that relate to body parts. Skeletal pooling operation allows information from long-distance joints to flow towards each other. Moreover, Xu et al. stack several pooling and unpooling layers in the network to facilitate multi-scale feature learning. They also utilize the pre-aggregation strategy explored by [Liu+20] for weight unsharing in order to extract and aggregate features specific to each joint in the skeleton. This combination achieves good localization accuracy and surprising generalization capabilities [XT21]. Figure 3.3 visually demonstrates how pooling and unpooling are carried out in this method. One potential downside of this method is that it requires more parameters than previously discussed approaches because of repeatedly stacked pooling and unpooling operations.

3.1.2 Transformer Based Methods

The *Transformer* proposed in [Vas+17] is a recent and popular architecture based on the attention mechanism. The attention mechanism in the Transformer serves the purpose of modeling interactions and relationships between inputs to the model. The attention scheme used in [Vas+17] uses a similarity measure between special feature vectors called *query* and *key* to determine the degree of relationship between two inputs. These vectors are computed for each input during training. Using these vectors, the Transformer builds an affinity matrix establishing the connectivity between inputs, much like GCNs. This method is more flexible than GCNs since it conditions the relationship between inputs to the inputs themselves, whereas, in GCNs, the relationships are modeled with an adjacency matrix which does not usually depend on the inputs.

For 3D HPE, this means that long-distance spatial relationships between joints of the skeleton can be modeled. This is especially useful when modeling temporal relations in the input, which is discussed in the next chapters. The predictive power of transformer-based architecture is also high, leading to accurate predictions. However, using transformer-based methods come at a cost. Namely, training these architectures requires more time and computational power. Models can potentially overfit the training data if the data does not contain enough variety [ZWT22; Zha+22; Zhe+21]. There is also a risk of not exploring the prior information of skeletal connections in the human body due to the freedom of relationships between inputs in the transformer.

GraFormer presented by Zhao et al. in [ZWT22] combines a transformer with a GCN in order to prevent the issue mentioned above from arising. To explore natural skeletal relationships between inputs, the authors use stacked layers of self-attention followed by a graph convolutional network with learnable adjacency matrices. The authors also utilize Chebyshev convolutional blocks that utilize fixed adjacency matrices modeling the exact skeletal structure of the body. The goal of self-attention is the extraction of long-distance relationships using similarity between joint features, while the GCN exploits skeletal priors. Particularly, the authors notice that the attention mechanism in GraFormer increases the performance when the distance between adjacent joints is larger [ZWT22].

3.1.3 Summary

Overall, the adaptation of graph convolutions in 3D HPE has been shown to increase the performance of models. Firstly, MLP-based approaches [Mar+17] were adapted to use graph networks [Zha+19b], strengthening the exploitation of joint neighborhood relationships by models. Then, graph neural networks and their properties of a message passing between adjacent nodes and weight sharing were further studied and improved in [Liu+20]. Later, works by [ZWT22; BGK21] demonstrated that splitting the neighborhood of joints into separate groups and performing non uniform graph convolution per each group results in better performance with a lower amount of parameters. Making the adjacency matrices learnable also allowed the models to learn relationships outside of fixed skeletal priors. Xu et al. [XT21] explored the importance of multi-scale feature extraction and aggregation for generalization performance on unseen human poses. Lastly, [ZWT22] successfully incorporated attention-based transformers and combined them with GCN layers to create a model that can explore long distant spatial relationships in the skeleton alongside fixed geometric priors.

Despite the impressive performance that modern single frame-based models achieve, they still suffer from not utilizing the temporal continuity of input frames. This, as discussed in the previous chapter, leads to a rugged sequence of outputs, which is not feasible for some

use cases such as visual tracking. To fill this gap, the exploitation of motion priors for human pose estimation is performed in this thesis with the goal of finding a framework that is both accurate and efficient. Core ideas of extracting spatial features and relationships using GCNs are explored with an extension of hierarchical temporal processing in order to incorporate motion priors into 3D HPE.

3.2 Multi Frame Models

3.2.1 Spatio-Temporal Convolution Based Methods

VideoPose3D proposed by Pavllo et al. in [Pav+19] is one of the most important works for multi frame 2D to 3D lifting methods. They propose using a one dimensional temporal convolution operator that aggregates temporal features of joints and their time-backward and time-forward neighbors in a sequence. Dilated convolutions are used for temporal downsampling and a combination of long and short-term features in the sequence. That is, the aggregation with dilated convolution increases the receptive field of the model exponentially, allowing it to process long-term relationships from the entire sequence. From an intuitive point of view, the model first extracts local features from adjacent frames and then combines those features into global temporal features to be processed in the following layers.

This process is repeated until the desired receptive field is achieved. Figure 3.4 visually demonstrates the process. The authors also explore the idea of semi-supervised learning with reprojection loss and consistency constraints between input 2D poses and 3D output poses. Semi-supervised learning helps when the availability of labeled 3D poses is limited. Instead of using ground truth 3D poses for supervision, semi-supervised model projects predicted 3D pose back to the image space using camera parameters and perspective projection and carries out supervision via computing the loss between projected and input 2D poses as shown in figure 3.5. So, the authors also train a trajectory model which only predicts the location of the root joint since the position of the skeleton on the image space depends on it as well. Without it, the 3D skele-

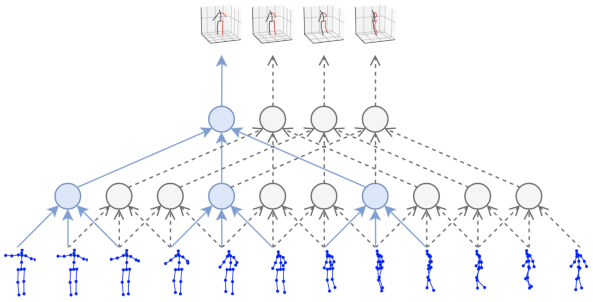

Figure 3.4: Visualisation of dilated convolutions in VideoPose3D. First, a temporal convolutional network is used to extract short-term features. Then, these features are combined and processed by the next layers. Thus, the network gains a large receptive field and processes the global temporal features from the whole sequence. Adapted from [Pav+19].

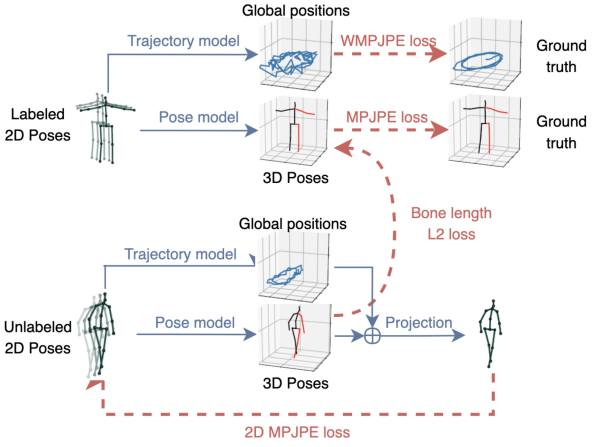

Figure 3.5: The semi-supervised approach with the inclusion of reprojection loss and the trajectory model. Trajectory and root relative joint locations are combined to project back to image space and perform supervision when ground truth data is scarce. Adapted from [Pav+19].

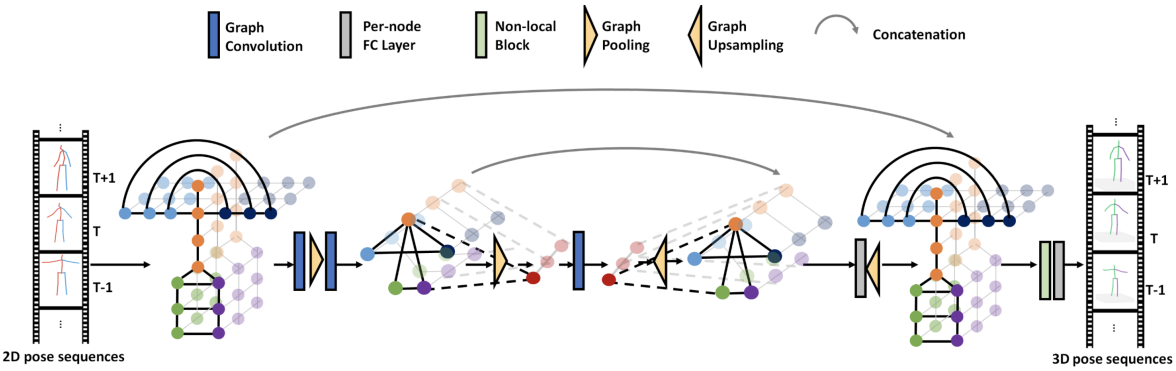

Figure 3.7: Visual demonstration of spatio-temporal GCN proposed by Cai et al. in [Cai+19a]. Temporal relationships between joints in a sequence of skeletons are demonstrated with connecting lines. The input graph is pooled using graph pooling to aggregate local features of body parts into global features. Unpooling is also performed to enable the interplay of local and global features. The colors of joints represent the body parts. Joints of the same color are pooled into one for aggregation. Adapted from [Cai+19a].

ton would always be projected to the center of the image with a fixed scale. The regression of the coordinates of the root joint suffers more from the depth ambiguity problem. Hence, the authors use a modified weighted loss that penalized the network less when the skeleton is further away from the camera. Overall, the authors show that it is possible to regress the global trajectory of the motion from an input sequence. Unfortunately, they do not provide details on the performance of the trajectory model. The approach presented in the later sections of this thesis also learns the global trajectory from the input sequence but does so using only one model that performs 3D HPE for all joints in the skeleton.

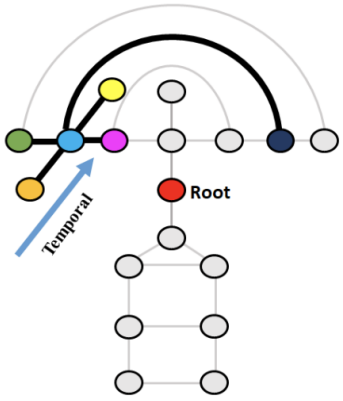

Figure 3.6: Graphical showcase of neighborhood groups used by Cai et al for. The neighboring nodes are divided into six classes according to their semantic meanings: center node (blue), physically-connected nodes including the one closer (purple) to and the one farther (green) from the skeleton root, indirect symmetrically related node (dark blue), time-forward node (yellow), and time-backward node (orange). Adapted from [Cai+19a].

The downside of this method is that it does not explore skeletal relationships within each frame. Each skeleton in the sequence is treated as an entire token by collapsing the coordinates of all joints into a single feature vector. Hence, unlike GCN or transformer-based methods, the interaction between individual joints is not explored by the model. Thus, to increase the performance of the model, the authors resort to using a high hidden dimension of 1024 which increases the computational complexity and the number of parameters required to run the model [Pav+19]. Furthermore, they train a separate trajectory model for semi-supervised learning, which doubles the computational cost. Works coming later improve on modeling interaction between joints in skeletons in the individual frame as well as the interaction between joints in the temporal dimension using more advanced architectures.

Spatial-Temporal Graph Convolutional Networks is an example of such an architecture

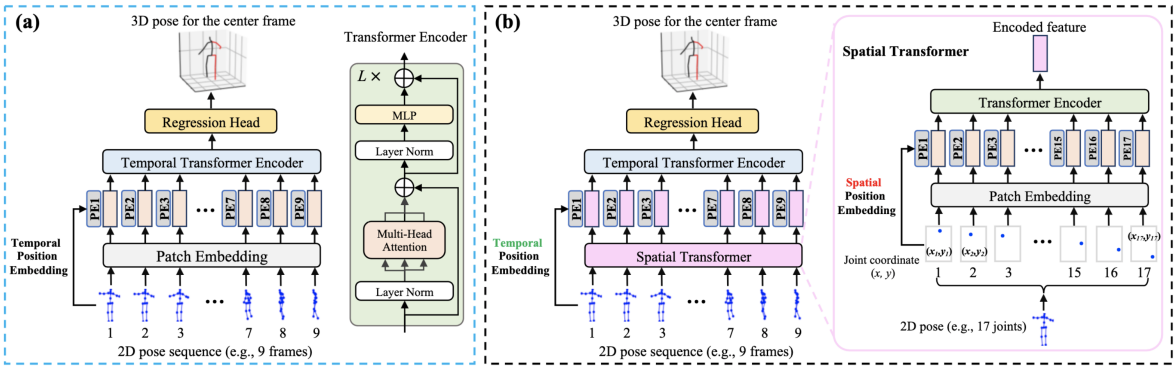

Figure 3.8: PoseFormer temporal transformer baseline and the spatial-temporal transformer. a) shows the temporal transformer baseline that embeds each skeleton as a whole without considering relationships between joints within each skeleton. b) demonstrates the spatial-temporal model which consists of three modules. A spatial transformer module for extracting features by considering joint correlations of each individual skeleton. A temporal transformer module for learning global dependencies of the entire sequence. A regression head module regresses the final 3D pose of the center frame. Adapted from [Zhe+21].

proposed by Cai et al. [Cai+19a]. They propose using spatio-temporal graph convolutions for multi-frame input 3D HPE. Cai et al. combine spatial skeletal pooling, similar to the one defined in [XT21] with spatio-temporal graph convolution. They model temporal relationships between inputs by combining the input sequence of 2D skeletons into a spatio-temporal graph where each joint is connected to its temporal neighbors. Figure 3.7 shows the architecture overview with hierarchical pooling and spatio-temporal relations. Moreover, Cai et al. use non-uniform graph convolutions by splitting the neighborhoods of joints into 6 groups, as shown in 3.6. Namely, 1) self-links of the nodes; 2) physically connected joint closer to the pelvis; 3) physically connected joint further from the pelvis; 4) symmetrically related joints (i.e., left shoulder to right shoulder or left hip to right hip); 5) time-forward connection to the same type of joint in the sequence and finally; 6) time-backward connection to the same type of joint in the sequence and finally. This kind of meticulous split of the graph enables the model to extract features specific to each neighbor group, increasing the positional accuracy of the model. The authors conduct experiments and note that dividing graphs into groups according to the semantics of joints boosts accuracy. The same conclusion is made regarding the number of input frames. It is demonstrated that increasing the number of input frames boosts the model's performance, showing that temporal relationships are a crucial cue for 3D HPE tasks. A limitation of this framework is that it considers sequences with lengths of up to 7 frames. As shown in figure 3.10, videos are typically shot at high frame rates, and consecutive frames typically contain similar poses. So, 7 input frames are insufficient to capture long-range temporal cues because of limited motion and receptive field.

3.2.2 Transformer Based Methods

PoseFormer is the first fully transformer-based architecture that tackles multi-frame 3D HPE. It was proposed by Zheng et al. in [Zhe+21], which is built following the pipeline of ViT [Dos+21]. Similarly to dilated convolutional model in [Pav+19], the authors first develop an architecture that treats each input skeleton as a single feature vector. It takes an input of a sequence of 2D joint detections for each frame and embeds each of them into a feature vector before processing by a temporal transformer. This leads to the network capturing only global skeletal features from each frame. Hence, joint-to-joint relationships are not utilized

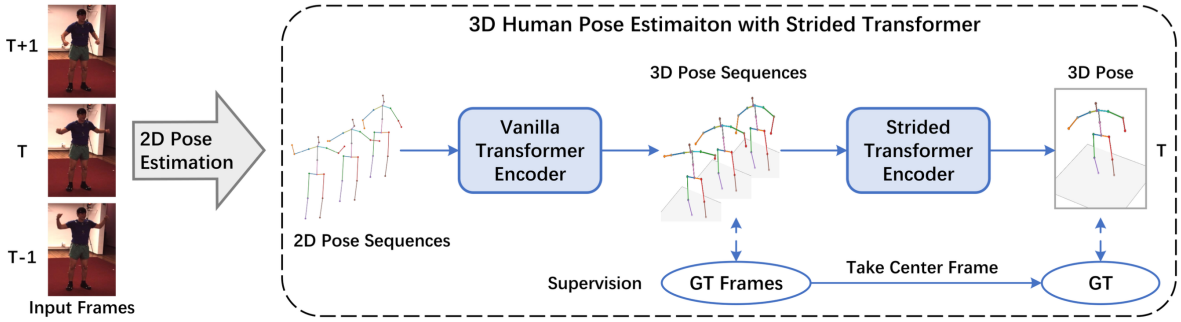

Figure 3.9: The full framework of Strided Transformer. 2D detections of all frames are first processed by a regular transformer encoder. It regresses intermediate 3D poses for all input frames. Strided transformer module then refines intermediate predictions to achieve the final prediction for the middle frame. Adapted from [Li+23].

in this baseline model. The architecture of this baseline is shown in figure 3.8 a).

The authors then propose a framework that performs feature extraction at two levels: 1) joint to joint level in each frame separately; 2) frame to frame level across the temporal dimension. As shown in figure 3.8 b), the model consists of 3 stages. First, each skeleton in a frame is treated as a sequence of 2D joint locations where each 2D joint is considered as a token. Similar to GraFormer [ZWT22], this stage models skeletal relationships within a single frame. The strength of the relationships between each pair of joints is determined by the self-attention mechanism instead of a pre-defined adjacency matrix. At the end of spatial feature extraction, each 2D skeleton is embedded into a feature vector that already contains local joint-to-joint correlation information due to the self-attention processing. Then, a temporal transformer is used to model correlations between feature vectors of each frame in the sequence before applying a regression head that predicts 3D joint locations for the target (middle) frame. This interplay of spatial and temporal transformer modules ensures that both spatial and temporal contexts are utilized by the model.

Overall, the method demonstrates that even with a low number of frames as input, the accuracy of the reconstructed 3D pose is impressive. The model also performs well when generalizing to unseen datasets such as [Meh+17]. However, the model has about 10M parameters regardless of the input size. Also, the transformer architecture requires larger computational power since the complexity grows quadratically with the size of the input [Has+22].

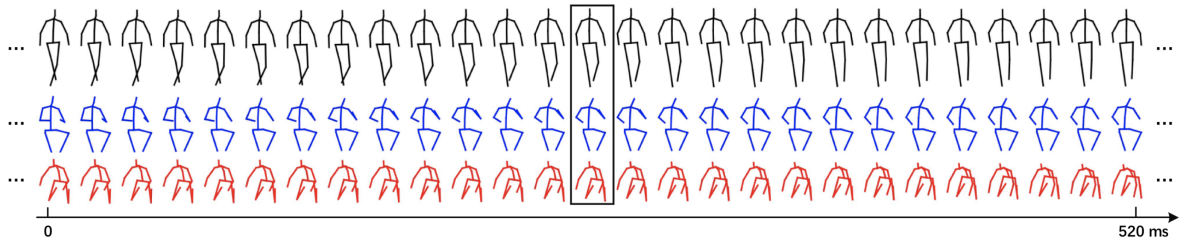

Figure 3.10: Showcase of 27 frame long cuts from Human3.6M dataset with 50Hz cameras. The rectangle indicates the middle frame of the video. As can be seen, consecutive frames contain an abundance of redundant information since nearby poses are almost identical due to high framerate cameras. Adapted from [Li+23].

StridedTransformer presented in [Li+23] attempts to fix the performance drawbacks of transformer architectures by gradually merging features of nearby skeletons, thus making the input smaller as the network goes deeper. As consecutive poses in a sequence contain redundant information (see figure 3.10) strided architectures can still extract sufficient in-

formation without having to process the whole data. Hence, the authors present a modified *Strided Transformer* module which uses strided convolutions in order to reduce the temporal dimension of the input. The architecture consists of 2 stages and has 2 points of supervision. First, a regular transformer encoder is applied to capture global features across the entire sequence. This module predicts 3D poses for all input frames for an intermediate supervision. Then, a strided transformer module progressively processes the output 3D poses from the previous module and progressively shrinks the number of inputs. Finally, a supervision for the middle target frame is performed. Intermediate supervision for all inputs forces the network to output temporally smooth and coherent poses. The strided transformer block can also be thought of as a refinement stage that refines the intermediate 3D poses. The complete architecture is displayed in figure 3.9. This approach decreases the computational complexity of the task. Moreover, intermediate supervision for the whole sequence provides richer signal to the first encoder leaving the task of refinement to the second module making. This boosts the performance of the model [Li+23]. Despite the performance improvements, the model has significant overhead due to the transformer architecture [Li+23]. This opens room for more research on efficient models for multi-frame 3D HPE.

Mixed Spatio-Temporal Encoder (MixSTE) proposed by Zhang et al. [Zha+22] builds upon the regular transformer encoder used in previous works. However, the authors argue that encoding the whole frame into a token and modeling the temporal context around that token is insufficient, as different joints in the body have different movement patterns. Some joints are more flexible than others, which necessitates using encoders that treat each joint as a separate feature token to fully model its motion in the sequence. MixSTE follows a sequence-to-sequence framework that predicts a 3D pose for all frames, not a single (target) frame.

Much like PoseFormer [Zhe+21], MixSTE has two types of blocks, spatial transformer block, and temporal transformer block. The striking difference between these approaches is how these blocks are used. While PoseFormer uses them in a sequential manner, MixSTE interweaves them so that the model extracts spatial-temporal features repeatedly in loops instead of sequentially one after the other. As shown in figure 3.11, first, a spatial transformer models interconnections between joints in each frame, then temporal transformer models interconnections of each type of joint across the sequence. This decoupling of spatial and temporal modules and their interplay ensures that the model extracts rich features for each joint individually.

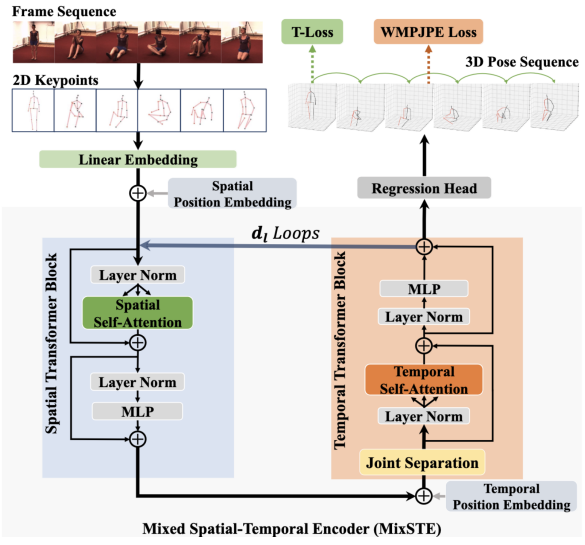

Figure 3.11: Overview of the MixSTE sequence to sequence architecture. Spatial and Temporal transformer encoders are repeatedly looped for rich spatial and temporal feature extraction and modeling of interactions between joints. The model is supervised with temporal loss function and joint error loss function for smoothness and accuracy. Adapted from [Zha+22].

Another key design choice is the supervision of the model with temporal loss functions that encourage the model to produce smooth motion. This regime of supervision enables the model to output realistic sequences unlike single-frame based methods [Zha+22].

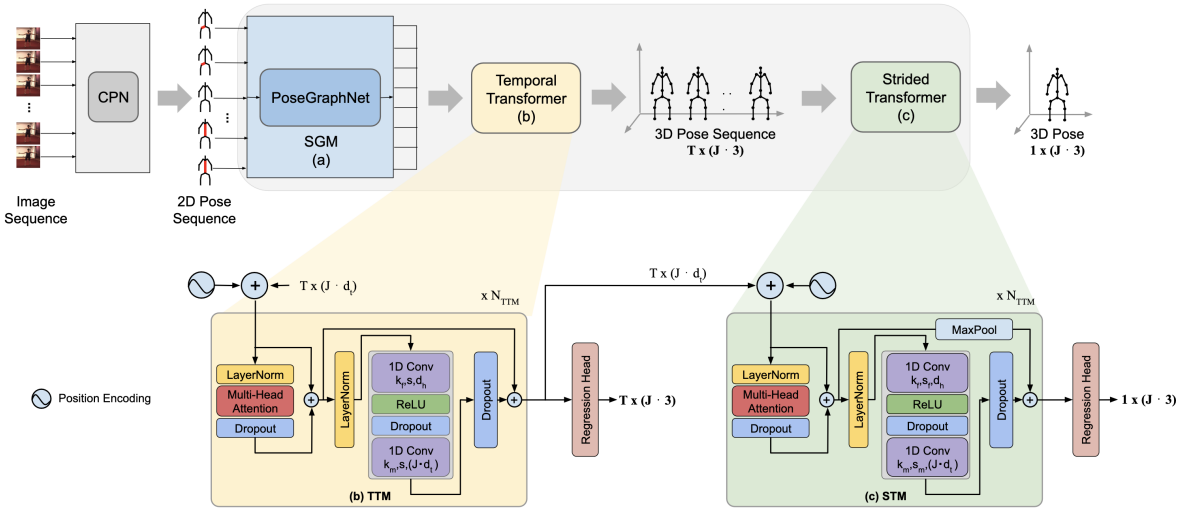

Figure 3.12: Illustration of the StridedPoseGraphFormer [Ban+23] architecture. The top diagram shows the overall pipeline of the framework predicting 3D joint positions of the middle frame from a 2D pose sequence. The input 2D pose extracted using CPN [Che+18] is augmented by zeroing joints for occlusion robust training. The model consists of (a) **Spatial Graph Module (SGM)**, (b) **Temporal Transformer Module (TTM)**, and (c) **Strided Temporal Module (STM)**. The model captures the spatial information via SGM, and the temporal context through TTM and STM to overcome occlusion. The bottom diagram further elaborates on parts of the used architecture. Adapted from [Ban+23]

StridedPoseGraphFormer proposed by Banik et al. in [Ban+23] combines a graph neural network with two transformer modules to develop an occlusion robust 3D pose estimation model. StridedPoseGraphFormer consists of 3 core modules, a) Spatial Graph Module (SGM), (b) Temporal Transformer Module (TTM), and (c) Strided Temporal Module (STM). The model extracts meaningful spatial relationships via SGM. Then, TTM predicts intermediate 3D poses for all input frames. Finally, the STM module uses a strided transformer to gradually shrink the input and predict the 3D pose for the middle frame. To achieve an occlusion robust model, StridedPoseGraphFormer was trained using a synthetic data augmentation technique that masks joints in the input 2D pose sequence. Hence, the model has to learn to recover accurate pose from spatial-temporal data despite missing joint locations. Thus, the model becomes robust to noise introduced by inaccurate 2D joint detectors.

3.2.3 Summary

Adaptation of 3D HPE methods to process multi-frame input began with utilizing temporal convolutions. VideoPose3D [Pav+19] used dilated temporal convolutions to extract and aggregate short-term temporal features into global features to accurately predict the 3D pose of the middle frame. This architecture shows that temporal dimension and motion prior contain valuable information for 3D HPE as the results significantly improved when compared to single frame-based methods. Later works expanded the architecture and included spatial and temporal modules to model intra-frame skeletal connectivity and inter-frame temporal connectivity between joints. Spatio-temporal graph convolution by Cai et al. [Cai+19a] used spatio-temporal graph convolutions which extended regular GCNs with the inclusion of temporal connectivity in the adjacency matrix. The authors also incorporated a hierarchical structure for spatial feature extraction with graph pooling and unpooling. Their experiments showed that increasing the receptive field of the model by using more frames in input re-

sulted in increased performance. However, the framework was hindered by a relatively low receptive field of 7 frames.

Follow-up works used transformer architectures to model local and global temporal relationships in longer sequences. PoseFormer [Zhe+21] combined spatial and temporal transformers for joint-to-joint and frame-to-frame connectivity modeling. Strided Transformer [Li+23], based on the idea that consecutive frames in a video are similar to each other, proposed to gradually merge nearby temporal features for efficiency. They used a temporal transformer to estimate rough 3D poses for all input frames, then refined them with a strided transformer. MixSTE [Zha+22] is a sequence-to-sequence model that proposed to model each joint separately while looping between spatial and temporal transformers to combine both features, increasing performance compared to prior works. StridedPoseGraphFormer [Ban+23] combines a graph convolutional network with temporal transformers to achieve an occlusion robust model. During training, input joint locations are synthetically set to 0 to simulate occlusion. Such a training regime allows the network to overcome noise introduced by 2D joint detectors such as [Che+18].

Though transformer-based methods achieve state-of-the-art results in multi-frame 3D HPE, they are limited by the computational power required. Moreover, these architectures are hard to train and take longer to converge. Thus, to improve the efficiency of 3D HPE models with little compromise, this thesis explores a spatio-temporal GCN-based method with a larger receptive field that extracts multi-scale spatial and temporal features for a sequence-to-sequence 3D HPE.

Chapter 4

Methodology

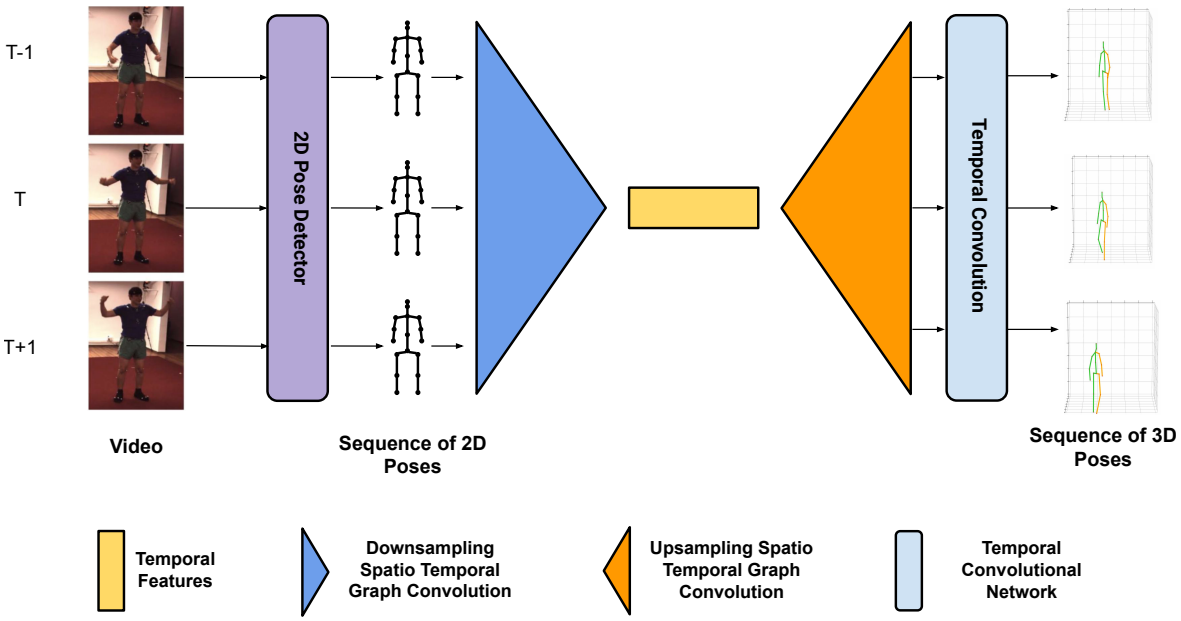

Figure 4.1: Overview of the proposed spatio-temporal graph convolution-based sequence to sequence architecture, TGraphNet. TGraphNet takes as input a sequence of 2D poses which are obtained from a video using a 2D human pose detector. Input 2D poses are processed via spatio-temporal graph convolution and gradually aggregated across temporal dimensions in the downsampling stage to obtain multi-scale temporal features. Gradual aggregation allows the model to extract global features from the whole sequence. Downsampled features are then procedurally upsampled and combined with multi-scale temporal features obtained during downsampling. Finally, a sequence of 3D poses is predicted using a temporal convolution that processes combined multi-scale features to generate a smooth sequence of 3D human poses.

This thesis explores the problem of efficient and accurate 3D pose sequence estimation using spatial-temporal graph neural networks. Approaches proposed in this work aim to produce smooth and continuous sequence of 3D poses by exploring both spatial and temporal contexts of relationships between the inputs. Two models, TGraphNet and TGraphNet (traj) are proposed and analyzed. TGraphNet, following the traditional 2D to 3D lifting paradigm, only estimates root joint relative 3D pose. TGraphNet (traj), in addition to root relative 3D pose also estimates the global position and motion of the root joint. This section describes both approaches and their components in detail.

4.1 Overview of Spatial-Temporal GCN Based Approach

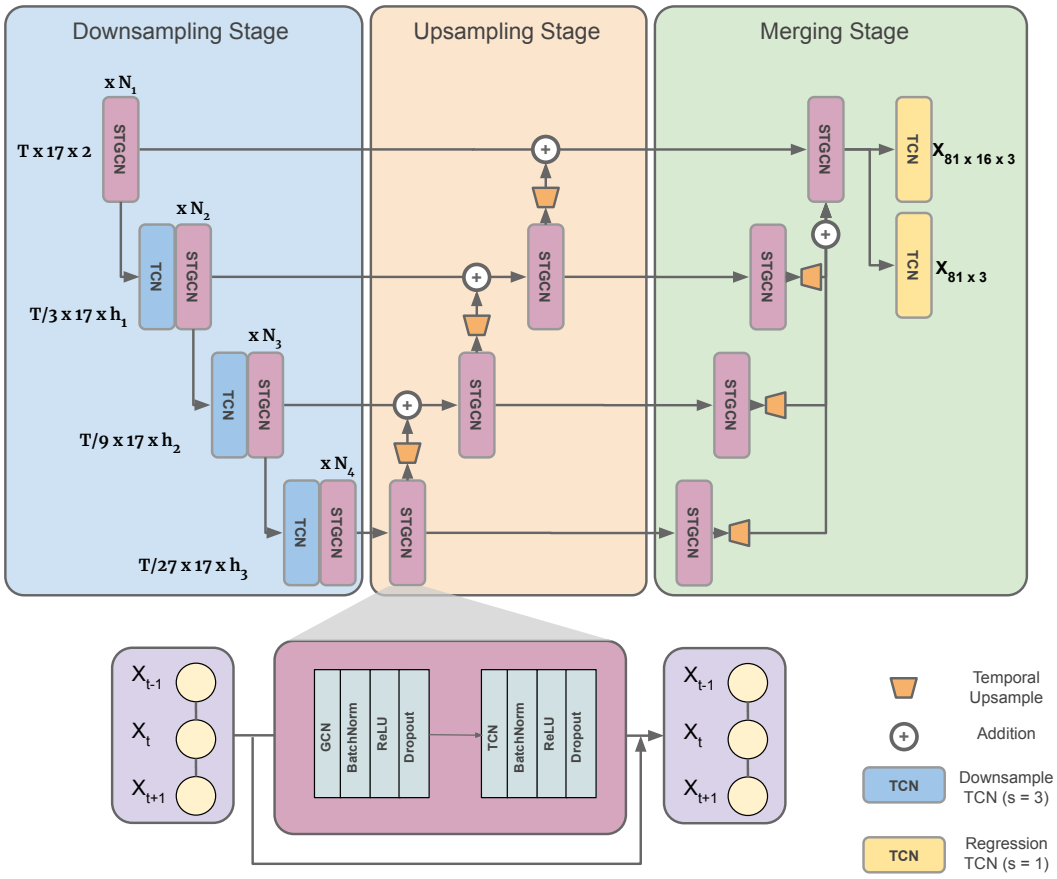

Figure 4.2: The full architecture diagram of TGraphNet and TGraphNet (traj). The model consists of 3 stages, downsampling, upsampling, and merging. In the downsampling stage, the input sequence of 2D skeletons is processed by spatio-temporal graph convolution and downsampled with temporal convolution for multi-scale feature extraction. Hidden dimensions h and the number of STGCN blocks N are depicted in the diagram. In the upsampling stage, aggregated temporal features are upsampled and added to hierarchical features to preserve detail. Finally, the merging stage combines temporal features from all scales before regressing root relative 3D poses. TGraphNet (traj) additionally employs a temporal convolution that regressed the global trajectory.

The proposed Spatio-Temporal GCN-based architecture, named *TGraphNet*, estimates the 3D pose sequence for the input video by exploiting multi-scale spatial and temporal features. As shown in the architecture overview in figure 4.1, the model takes as input a sequence of 2D poses obtained by running an off-the-shelf 2D pose estimator on the videos. Specifically, Cascaded Pyramid Networks (CPN) presented in [Che+18] is used. The input sequence of 2D skeletons is arranged into a sequence of short-term spatio-temporal graphs to be processed by a spatio-temporal graph convolutional network (STGCN). Graph representation models the relational inductive bias in the human body by having the connections between joints mimic the natural connections in the human skeleton. To enable flexible connectivity in the network and avoid the hindrance of weight sharing (see section 2.2.4), the graph is split into multiple neighborhood groups according to the semantics of joints following [BGK21; Cai+19b]. Adjacency matrices are made adaptive so that the model explores connectivity beyond natural skeletal structure. Additionally, temporal relationships between joints and their past and future instances are modeled through temporal convolutions. To capture spatio-temporal relationships in long input sequences, a U-shaped network architecture with several stages is

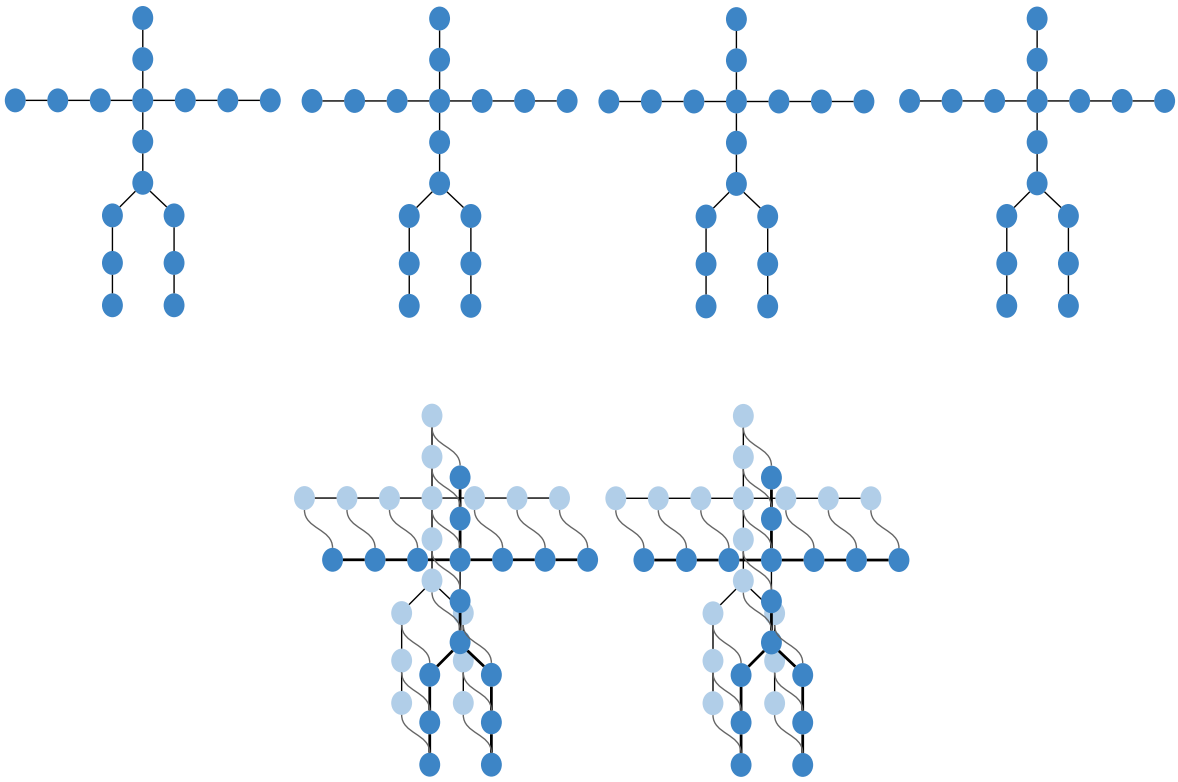

Figure 4.3: A visual showcase of how a sequence can be combined into smaller spatio-temporal graphs. **Top** shows a sequence of 4 human skeletons is shown. It is split into groups of two adjacent skeletons and temporal connections are established for nodes of the same type. **Bottom** shows the combination of skeletons into spatial-temporal graphs. Gray lines denote temporal connections and black lines denote spatial links.

employed, as demonstrated in figure 4.2. In the downsampling stage, temporal pooling is applied to gradually merge short-term features into more global, long-range temporal features. This enables the network to have a large receptive field, thus enabling it to extract long-range features from the input. The upsampling stage is used to gradually recover the temporal resolution. At each scale, downsampled features are added to the upsampled features via skip connections to preserve details. Finally, the merging stage combines all the features from all timescales before 3D poses are estimated. The final 3D pose sequence is obtained by running a temporal convolution to produce a smooth and continuous 3D sequence. One regressing temporal convolution is used to obtain root joint relative 3D pose, while the other is used to regress the global trajectory or the motion of the skeleton.

4.2 Formation of The Spatio-Temporal Graph and Feature Extraction

Modeling the whole input sequence as a single large spatial-temporal graph is inefficient since the size of the adjacency matrix grows quadratically with respect to the length of the sequence. To cope with this hindrance, a sequence is partitioned into smaller short-term spatial-temporal graphs. Figure 4.3 visually demonstrates this process. A sequence of length 4 is combined into a sequence of spatial-temporal graphs of length 2 where each joint is connected to its adjacent instance. The STGCN block processes each short-term spatio-temporal graph, and then the extracted features are combined, leading to a large receptive field and avoiding infeasible computational complexity. In this approach, the time window of short-term graphs is set to 3, meaning that each 3 neighboring 2D poses are combined into a single

graph. Formally, let $X_t = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T \mid \boldsymbol{x}_i \in R^{17\times 2}, i = 1, \ldots, T\}$ be the sequence of inputs, then the partitioning of the whole sequence into a sequence of short-term spatio-temporal graphs would be
$G_t = \{[\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3], [\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6], \ldots, [\boldsymbol{x}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{x}_T] \mid \boldsymbol{x}_i \in R^{17\times 2}, i = 1, \ldots, T\}$ where each skeleton is paired with its past and future instances. Each graph in this sequence can be formalized as an attributed graph $g_{2D} = (V, E, X)$ where E is the set of edges, $V = \{v_{tj} \mid t = 1, \ldots 3; j = 1, \ldots, 17\}$ are the joints and $X = \{\boldsymbol{x}_{tj} \in R^2 \mid t = 1 \ldots 3, j = 1 \ldots 17\}$ are the 2D locations of each joint in each frame of the sequence. Now that the graph also contains temporal dimension, the set of edges can be partitioned into spatial edges and temporal edges. Spatial connectivity defines the natural skeletal connections of joints in each skeleton of each frame, while temporal edges define the connectivity of joints with their past and future instances in nearby frames. Applying graph convolutions in this setup of short-term graphs extracts both spatial and temporal features from the graph for all joints across all frames. Aggregating these features across the temporal dimension achieves a combined spatial-temporal representation of each joint and reduces the dimensions. Formally, let $G_h = \{[\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3], [\boldsymbol{h}_4, \boldsymbol{h}_5, \boldsymbol{h}_6], \ldots, [\boldsymbol{h}_{t-2}, \boldsymbol{h}_{t-1}, \boldsymbol{h}_T] \mid \boldsymbol{h}_i \in R^{17\times d_h}, i = 1, \ldots, T\}$ be the sequence of spatio-temporal graphs after feature extraction with a graph neural network. Here, $\boldsymbol{h}_i$ are the feature vectors for all joints in a single skeleton, and d_h is the hidden dimension. Then, if features of each individual spatial-temporal graph $[\boldsymbol{h}_{t-1}, \boldsymbol{h}_t, \boldsymbol{h}_{t+1}]$ in the sequence are combined, $\tilde{\boldsymbol{h}} = f(\boldsymbol{h}_{t-1}, \boldsymbol{h}_t, \boldsymbol{h}_{t+1})$, temporal pooling is achieved. In this equation, f is some aggregating operation such as average pooling or temporal convolution, and $\tilde{\boldsymbol{h}} \in R^{17\times d_h}$ are the pooled features for all 17 joints. For the proposed methods TGraphNet and TGraphNet (traj), temporal convolution with a stride of 3 is used for more flexible temporal downsampling. This is demonstrated in the downsampling stage of figure 4.2, which also shows the input and output dimensions. This operation of combining inputs into spatio-temporal graphs with window 3 and pooling is repeated to achieve a large receptive field, as demonstrated in the figure.

4.3 Adjacency Matrix and The Neighborhood Groups

Since the graph convolution in the proposed architecture interacts with spatio-temporal graphs, the adjacency matrix that is used in GCN formulation also contains temporal links. Thus, the spatio-temporal adjacency matrix can be defined as $\boldsymbol{A} \in R^{TJ\times TJ}$ where T is the length of the temporal window, and J is the number of joints. In this case, J is set to 17, and T is set to 3. It can be conveniently represented in a block form where the diagonal submatrices represent spatial connections between joints in the same frame, and off-diagonal matrices contain temporal connections.

$$\left[\begin{array}{c|c|c} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \boldsymbol{A}_{13} \\ \hline \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \boldsymbol{A}_{23} \\ \hline \boldsymbol{A}_{31} & \boldsymbol{A}_{32} & \boldsymbol{A}_{33} \end{array}\right] \tag{4.1}$$

Here, each sub adjacency matrix $A_{t_1 t_2} \in R^{17\times 17}$ where $t1, t2 \in \{1, 2, 3\}$ contains connections between joints of frames t_1 and t_2. Thus, the multiplication of the graph joint attributes with this matrix performs message passing both across temporal and spatial dimensions. To see this, let $\boldsymbol{X} = \left[\begin{array}{c|c|c} \boldsymbol{X}_1 & \boldsymbol{X}_2 & \boldsymbol{X}_3 \end{array}\right]^{\mathrm{T}}$ be the concatenated 2D joint locations in each frame in the spatio-temporal graph with temporal window of 3. $\boldsymbol{X}_1 \in R^{17\times 2}$, $\boldsymbol{X}_2 \in R^{17\times 2}$, $\boldsymbol{X}_3 \in R^{17\times 2}$ are the joint locations of the first, second, and third frames in the sequence. Multiplication of the block adjacency matrix with concatenated joint locations yields

$$\left[\begin{array}{c|c|c} A_{11} & A_{12} & A_{13} \\ \hline A_{21} & A_{22} & A_{23} \\ \hline A_{31} & A_{32} & A_{33} \end{array}\right] \left[\begin{array}{c} X_1 \\ X_2 \\ X_3 \end{array}\right] = \left[\begin{array}{c} A_{11}X_1 + A_{12}X_2 + A_{13}X_3 \\ A_{21}X_1 + A_{22}X_2 + A_{23}X_3 \\ A_{31}X_1 + A_{32}X_2 + A_{33}X_3 \end{array}\right] \tag{4.2}$$

where each part of the resulting combined vector includes a message passing within the frame itself and a message passing between joints in the neighboring frames that are connected to each other.

For instance, consider the first block in the combined output vector. The term $A_{11}X_1$ is responsible for spatial message passing of joints within the first frame and $A_{12}X_2 + A_{13}X_3$ computes message passing of joints in the first frame with joints in the next two frames. Now, to perform a graph convolution, both spatial and temporal submatrices need to be initialized to be used in GCNs.

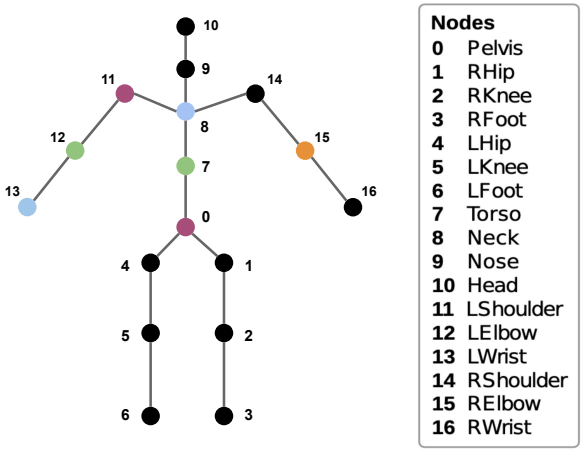

Figure 4.4: Representation of the skeleton as a graph with separated neighborhood groups. Joints in the skeleton are numbered, and their names, according to Human3.6m [Ion+14] are shown to the right of the skeleton. Example neighborhood relationships are shown for *Torso* and *LElbow* joints. Green represents the self link, magenta shows the parent link, light blue shows the child link, and orange shows the symmetric link. Note that some joints have no symmetric links.

4.3.1 Spatial Connections

As mentioned in chapter 2, applying a uniform feature transformation with a single shared weight matrix for all joints in the graph is too restrictive and leads to poor performance [XT21; BGK21; Cai+19b; Liu+20]. Hence, the joints in the graph are partitioned into neighborhood-specific groups. Each group shares a separate feature transformation weight matrix. This allows more flexible feature extraction per each neighborhood group. Specifically, joints are separated into 4 neighborhood groups where only selected types of connections are present. Namely, these groups are: a) Self, b) Parent (closer to the root joint), c) Child (further from the root joint), and d) Symmetric indirect connections. Figure 4.4 visually demonstrates different neighborhood links for two joints. Thus, 4 block adjacency matrices as defined in equation 4.1 are constructed where the diagonal sub matrices holding the spatial links between joints are defined according to the neighborhood group. Moreover, each adjacency matrix is adaptive, meaning that the model is free to learn connectivity outside of the natural skeletal structure. It is worth noting that even though only spatial connections are split into groups, a whole block matrix including temporal sub matrices is created allowing the network to learn different temporal links per each group. Additionally, adjacency matrices are degree normalized so that each neighbor contributes equally during message passing.

4.3.2 Temporal Connections

Temporal connections on the block matrix are initialized to zeros in the beginning. This is done because the spatio-temporal graph convolution block in the architecture already has a temporal convolutional module that extracts temporal features from adjacent joints. However, since the adjacency matrix in the network is learnable, the network has a chance to

learn temporal connections for graph convolutions as well. Moreover, since it performs non-uniform convolution with 4 neighborhood adjacency matrices and neighborhood-specific kernels, the network has a chance to learn separate temporal links per each group. In the next chapter, the learned adjacency matrices are visualized and analyzed after training is performed.

Overall, 4 block diagonal adjacency matrices are used for each neighborhood group. Spatial connections are initialized according to neighborhood groups depicted in figure 4.4. Temporal connections in each matrix are initialized to zeros which may be altered during the training process by the model. 4 adaptive adjacency matrices are used to perform non-uniform graph convolution on the input spatio-temporal graphs in order to extract rich, connectivity-specific features.

4.4 Network Modules

With the formulation of spatio-temporal graph sequence and neighborhood groups covered, network modules and the architecture can be covered in detail. As it can be seen in figure 4.2, TGraphNet is composed of 3 main stages, downsampling, upsampling, and merging. Each of these stages relies on the STGCN block, which is the core of the architecture. In the downsampling stage, STGCN is used to extract short-term features then strided convolutions are used for aggregation and downsampling, which can be seen in figure 4.5. Bilinear upsampling operation is used in upsampling and merging stages to recover the temporal dimension in order to combine features of different scales. These blocks are discussed in detail in this section.

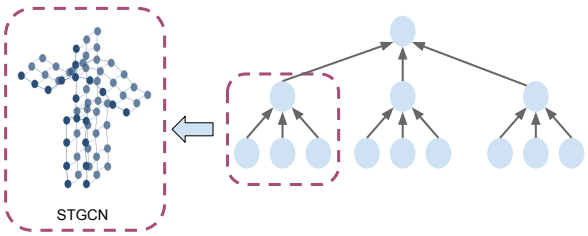

Figure 4.5: Visualization of short-term feature extraction and combination in the STGCN block. An STGCN applies a spatio-temporal convolution with a temporal window of 3. Then, strided convolution combines features to be processed in the next layer. In this case, a sequence of 9 inputs gradually gets downsampled to size temporal size 1 via repeated STGCN and temporal downsampling.

STGCN Block: STGCN block is shown in the architecture diagram in figure 4.2. It consists of a graph convolutional layer followed by batch normalization [IS15] for stable training and ReLU non linearity. Dropout [Sri+14] is used for regularization to prevent overfitting. Then, a temporal convolution is applied, which is again followed by batch normalization, ReLU, and a dropout layer. In the end, input features are added to the output of the STGCN block for residual skip connections, which enable faster convergence and prevent vanishing gradients [He+16]. The input to the STGCN is a sequence of spatio-temporal graphs
$G_t = \{[\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3], [\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6], \ldots, [\boldsymbol{x}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{x}_T] \mid \boldsymbol{x}_i \in R^{17\times 2}, i = 1, \ldots, T\}$ as defined at the beginning of section 4.2. Each graph $[\boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}]$ in the input sequence consists of 2D joint locations in 3 consecutive input frames. A non-uniform spatio-temporal graph convolution with 4 neighborhood groups as defined in section 4.3 is applied to the input. At this stage, the model extracts short-term spatio-temporal features from each graph in the sequence. After normalization, non linearity, and dropout, the same short-term spatio-temporal graph is processed by a TCN with a stride of 1. The TCN applies a convolution on the joint features across the temporal dimension in order to extract temporal features. TCN is the regular convolution operation performed across temporal dimensions for each joint. The forward propagation rules for the STGCN block can be formalized in the following way.

$$\tilde{\boldsymbol{H}}^{l} = \sigma\left(\sum_{g \in G} \boldsymbol{A}_g \boldsymbol{H}^{l} \boldsymbol{W}_g^{l}\right)$$
$$\boldsymbol{H}^{l+1} = \sigma\left(\mathrm{TCN}(\tilde{\boldsymbol{H}}^{l})\right) + \boldsymbol{H}^{l} \tag{4.3}$$

Here, $\boldsymbol{A}_g \in R^{51\times 51}$ and $\boldsymbol{H}^{l} \in R^{51\times d_{h_l}}$ are the adaptive adjacency matrix for the neighborhood group g and the combined feature vector for all joints in the short term spatio-temporal graph. $\boldsymbol{W}_g^{l} \in R^{d_{h_{l+1}}\times d_{h_l}}$ is the weight matrix of layer l and neighborhood group g.

Temporal Downsampling Block: The temporal downsampling operation in the model is used to combine the features of short-term spatio-temporal graphs from the sequence G_t. It reduces the temporal dimension and increases the receptive field of the model. It is defined as a regular TCN but with stride equal to 3. A stride of 3 allows it to combine the features of consecutive joints from 3 frames. Figure 4.5 demonstrates short-term feature extraction and pooling. A sequence of 9 inputs is gradually aggregated with temporal downsampling with a TCN.

Temporal Upsampling Block: The upsampling operation increases the temporal dimension in the merging and upsampling stages. It is defined as a regular bilinear upsampling and has no learnable parameters. Upsampled temporal features are combined with the respective downsampled features in skip connections as shown in figure 4.2. This allows the network to preserve detail and fuse multi-scale features. In the merging stage, features from all temporal scales are upsampled to the original input dimension for multi-scale feature fusion before regressing the final sequence of 3D poses.

Regression TCN Blocks: After merging features from multiple time scales in the merging stage, regressing TCN blocks are utilized for 3D pose regression for all input frames. These are regular temporal convolutions with stride being set to 1. TCN is preferred over the linear layer because it operates in a temporal context. In total, there are 2 regression TCN blocks. One for regressing root relative joint locations and the other for regressing only the trajectory of the person in the input. These are separated because the scale of root relative joints and the global pose is different. When training only for root relative pose, the output of the second regression head is not used.

4.5 Loss Functions

TGraphNet is supervised with a combination of loss functions in order to produce a smooth and realistic output sequence of 3D poses. The commonly used mean per joint position error (MPJPE) loss is not enough for proper supervision as it does not penalize the network for inconsistencies in velocity and smoothness of the output sequence. Moreover, in the case of training for global trajectory as well, using a backprojection loss that penalizes inconsistencies between projected 2D locations of joints and ground truth 2D locations of joints becomes beneficial. All the loss functions that were used for supervising TGraphNet are listed below. Different combinations of losses were tested, and the results are discussed in the next chapter.

MPJPE Loss: MPJPE loss measures the average error between the estimated joint locations and ground truth joint locations. It enforces the model to output correct 3D poses. It is defined as

$$L_{MPJPE} = \frac{1}{J}\frac{1}{T}\sum_{i=1}^{J}\sum_{t=1}^{T}||P_{i,t} - P_{i,t}^{gt}||_2 \tag{4.4}$$

where $P_{i,t}$ represents the estimated position vector for joint i at time step t, $P_{i,t}^{gt}$ is the vector of 3D positions of the ground truth joints for the same joint and time.

Trajectory Loss: The loss for supervising global trajectory prediction is the same as MPJPE loss with the only difference being that it is weighted with the inverse of the depth of the root joint, Z_{P_t} [Pav+19].

$$L_{traj} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{Z_{P_t^{gt}}}||P_t - P_t^{gt}||_2 \tag{4.5}$$

Reprojection Loss: Reprojection loss measures the discrepancy between the estimated 3D pose and the observed 2D poses, encouraging the estimated poses to align with the observed 2D poses. The reprojection is calculated using the camera's intrinsic parameters. This loss is important for semi-supervised learning [Pav+19] and for eliminating inconsistencies between input and output poses. It is calculated as follows

$$L_{proj} = \frac{1}{J}\frac{1}{T}\sum_{i=1}^{J}\sum_{t=1}^{T}||\hat{P}_{2D_{i,t}} - P_{2D_{i,t}}^{gt}||_2 \tag{4.6}$$

where $\hat{P}_{2D_{i,t}}$ and $P_{2D_{i,t}}^{gt}$ are the reprojected joint locations of predicted 3D joints and ground truth 2D joint locations at time t.

MPJVE Loss: Mean per joint velocity error (MPJVE) loss provides a quantitative measure of the accuracy of the estimated joint velocities. Joint velocities refer to the rate at which joints move in the 3D pose sequence. High MPJVE means that the velocity of joints in the ground truth sequence is significantly different from the velocities of the joints in the predicted 3D sequence. Supervision with MPJVE loss encourages the network to output realistic motions of joints. It is computed as follows

$$\begin{aligned} L_{MPJVE} &= \frac{1}{J}\frac{1}{T-1}\sum_{i=1}^{J}\sum_{t=1}^{T-1}||V_{i,t} - V_{i,t}^{gt}||_2 \\ V_{i,t} &= P_{i,t} - P_{i,t-1} \end{aligned} \tag{4.7}$$

where $V_{i,t}$ is the velocity or the 1st derivative of the position of joint i at time t. $P_{i,t}$ and $P_{i,t-1}$ are the locations of joint at time t and $t-1$.

TCL Loss: Temporal Consistency Loss (TCL) [Zha+22] measures the average error between the predicted joint positions at time t and the predicted joint positions at the subsequent time $t+1$ in order to encourage smooth and coherent motion sequences. It is computed as follows

$$L_{TCL} = \frac{1}{J}\frac{1}{T-1}\sum_{i=1}^{J}\sum_{t=1}^{T-1}||P_{i,t} - P_{i,t+1}||_2 \tag{4.8}$$

Motion Loss: Motion loss [Wan+20] uses a pairwise motion encoding to represent the trajectory of joints from time moment t_1 to time moment t_2. Then, the representations of trajectory for all joints are calculated for ground truth and predicted 3D sequences, and the average error between them is computed for supervision. Essentially, motion loss encourages

the trajectories of joints in the predicted sequence to be close to the trajectories of joints in the ground truth sequence. This loss is an alternative to MPJVE and TCL losses. The effectiveness of both is tested in the next chapter. Motion loss is computed as follows

$$\begin{aligned} L_{motion} &= \sum_{\tau \in \mathcal{T}} \sum_{i=1}^{J} \sum_{t=1}^{T-\tau} ||M_{i,t,\tau} - M^{gt}_{i,t,\tau}||_2 \\ M_{i,t,\tau} &= P_{i,t} \times P_{i,t+\tau} \end{aligned} \tag{4.9}$$

where $M_{i,t,\tau}$ is the motion encoding for time scale τ for joint i at time t. It essentially computes a descriptor of the trajectory of joint i from time t to $t+\tau$. In this case, cross product between vectors is used but any differentiable function computing similarity or difference can be used (ie. inner product). Several time scales $\tau \in \mathcal{T}$ are used to encode both long-term and short-term motions.
The final loss function for TGraphNet with only root relative pose output is formulated as

$$L = L_{MPJPE} + \lambda_{motion} L_{motion} + \lambda_{MPJVE} L_{MPJVE} + \lambda_{TCL} L_{TCL} \tag{4.10}$$

While TGraphNet (traj), which also predicts global trajectory is supervised with the addition of reprojection loss as well.

$$L = L_{MPJPE} + L_{traj} + \lambda_{motion} L_{motion} + \lambda_{proj} L_{proj} \tag{4.11}$$

λ in the loss equations above are the weights of the loss functions.

Chapter 5

Experimental Results

The following chapter presents the details of training and experimental results for TGraphNet and TGraphNet (traj) based on the architecture described in the previous chapter. One of the models, TGraphNet, follows the standard procedure and predicts only root relative 3D joint locations while the other, TGraphNet (traj), also predicts the 3D location of the root joint. These models and their respective results are discussed in separate sections. First, a description of the training setup is presented, then the results of experiments with ablation studies are discussed in more detail.

5.1 Experimental Setup

5.1.1 Datasets

TGraphNet is trained and evaluated on multiple state of the art datasets to measure the generalization capacity of the network. Below is a list of each dataset with a summary and its use in the scope of this thesis.

Human3.6M: Human3.6M [Ion+14] dataset is a standard popular dataset for training and evaluating the performance of 3D human pose estimation methods. It contains around 3.6 million images captured from 4 different viewpoints, and it includes 15 human subjects performing 17 different types of activities. The dataset is captured in an indoor scenario using marker-based motion capture. 2D joint locations in the image space are provided by back-projecting 3D joints using the perspective model. As per the standard protocol [XT21; Zhe+21; Zha+22], subjects 1, 5, 6, 7, and 8 are used for training, and subjects 9 and 11 for evaluation. The Human3.6M dataset, while versatile, does not fully encompass in-the-wild activities that humans perform. There is a large skew of the distribution towards walking and standing actions [Meh+17], which causes low performance of trained models on actions involving other actions such as sitting or crouching. Therefore, its potential for generalization performance evaluation is limited.

MPI-3DHP Dataset: MPI-3DHP dataset [Meh+17] captures 8 actors performing a variety of activities. The capture of ground truth 3D joint locations is done using a commercial markerless motion capture system. 3DHP dataset, in contrast to Human3.6M, also contains scenes that were captured outdoors. In addition, the test set of the 3DHP dataset has more variety in camera viewpoints making it a good choice for evaluating the generalization performance of 3D HPE models. This dataset is only used for evaluation purposes, and only the test set is

used. No training samples from this dataset are included in the training procedure.

MPI-3DPW Dataset: MPI-3DPW [Mar+18] is a multi-person 3D HPE dataset that captures people doing activities outdoors. This capture is performed in a completely in-the-wild setup using inertial measurement units (IMU) instead of motion capture. The dataset provides 3D human poses in SMPL [Lop+15] format. To obtain 3D joint locations, the joint regressor is employed that regresses 3D joint locations given the human body mesh in SMPL format. 2D joint locations are then obtained via back projection. This dataset is also only used for evaluating the model. It is worth noting that the camera setup is significantly different from that of Human3.6M, which introduces significant hindrances to generalization. These issues are discussed in detail later in the chapter.

5.1.2 Evaluation Metrics

Both models are evaluated for two criteria; accuracy of the estimations and accuracy of the motion of the predicted sequence. To evaluate the quality of the motion, the mean per joint velocity error is used. The definition can be found in equation 4.7. Mean per joint position error defined in equation 4.4 is used the measure the discrepancy between the predicted 3D joint locations and ground truth joint locations. This is often called *Protocol 1* (P1). Following other work [Zhe+21; BGK21; Cai+19a], *Protocol 2* (P2) is also evaluated which is the computation of MPJPE after *Procrustes Alignment* of ground truth and predicted 3D poses. To evaluate the performance of the global pose estimation task, MPJPE of predicted and ground truth locations of the Hip joint is computed separately from the rest of the joints.

For evaluation on the 3DHP dataset, commonly used *3D percentage of correct keypoints (3DPCK)* and *area under the curve (AUC)* metrics are used. 3DPCK is defined below

$$\mathrm{E}_{3DPCK} = \frac{1}{N}\sum_{j=1}^{N}[dist(j) \leq T] \tag{5.1}$$

where N is the number of test samples, $dist(j)$ is the Euclidean distance between the predicted and ground-truth 3D joint locations j, and T is the threshold which is set to 150mm, a commonly used value [Zhe+21]. Intuitively, 3DPCK counts the percentage of joints for which the errors is below the threshold. Thus, a joint is said to be correctly classified if it is within 150mm radius from the ground truth. The AUC is the area under the curve which is obtained by evaluating 3DPCK for thresholds ranging from 0 to 150mm.

5.1.3 Implementation Details

Data Processing

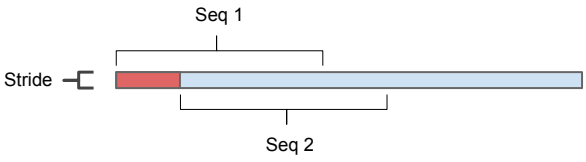

Figure 5.1: Display of the strided sequence batching. The gap between the start of the first and second sequences is the stride shown in red.

The input 2D poses for the model are extracted using CPN [Che+18] 2D pose detector. Most other works also report results using CPN detections for training, making it easier to compare with state of the art approaches. Input 2D joint locations are normalized to the range $[-1, 1]$ using height and width rescaling. To obtain an input sequence of 2D poses of specific length T, full

videos are organized into chunks of sequences of the said length. To do so, the strided batching strategy is used [Zha+22]. For each input frame in the video, past and future $\frac{T-1}{2}$ frames are concatenated into a single sequence of length T. If the sequence ends up being short, it is padded with the replicas of the first or last frame, depending on the side that lacks frames. However, this strategy leads to large overlaps between consecutive sequences, which results in redundant inputs. Hence, only each 5th such sequence is used for training. This process is visualized in figure 5.1. In addition, striding the input helps speed up the training procedure without compromising performance. For analyzing the effect of the receptive field and size of the input sequence on the performance of the model, $T = 9$, $T = 27$, $T = 81$, and $T = 1$ sequence lengths are selected for batching and training. Data augmentation is not used while training and testing the model.

Training Regimen

Both TGraphNet and TGraphNet (traj) models are trained only on Human3.6M dataset for 20 epochs using Amsgrad optimizer [RKK18] and a batch size of 64. The training is performed on a single NVIDIA RTX-4080 GPU. The learning rate is set to 0.001 initially and is decayed by a factor of 0.99 after every training epoch. The number of STGCN blocks is set as to $N_1 = 1$ and $N_2 = N_3 = N_4 = 2$ following the ablation study that is discussed later. Hidden dimension are set to $h_1 = h_2 = 128$ and $h_3 = 256$. These hyperparameters are visualized in figure 4.2. The probability of dropout is set to 0.1. For the baseline TGraphNet training, the length of the input sequence is set to $T = 81$. For an ablation study with a lower number of input frames, the downsampling TCN layers are removed to keep the correct dimensions. For $T = 27$, the first downsampling TCN is removed. For $T = 9$ input frames, the 2nd TCN downsampler is removed as well. For $T = 1$ input frames, no downsampling is performed whatsoever. The TGraphNet model for only root relative pose estimation is trained with MPJPE loss and Motion loss, ie. $\lambda_{MPJVE} = \lambda_{TCL} = \lambda_{proj} = 0$ while $\lambda_{motion} = 1$. Training of the TGraphNet (traj) model with global pose prediction follows a similar setup. The only difference is that the coefficient of motion loss for the root (Hip) joint trajectory is set to 0.001.

Inference Strategy

Since TGraphNet is a sequence-to-sequence (seq2seq) model, it is worth analyzing the positional error of the 3D pose for each input frame individually. Figure 5.2 shows the MPJPE per each 81 output frames of the model. It can be noted that 3D poses of middle frames are predicted with better accuracy. The first and last couple of frames have outlying positional errors compared to the rest. This phenomenon can be attributed to the larger temporal context for the middle frames. Moreover, temporal pooling in the downsampling stage of the model aggregates features towards the middle. So, to avoid additional error introduced by less accuracy in extreme frames, a sliding window inference strategy is employed where only the middle 31 predicted 3D poses are used via sliding the input window for 31 frames each time. The same strategy is used for TGraphNet (traj) as well. This way, more accurate

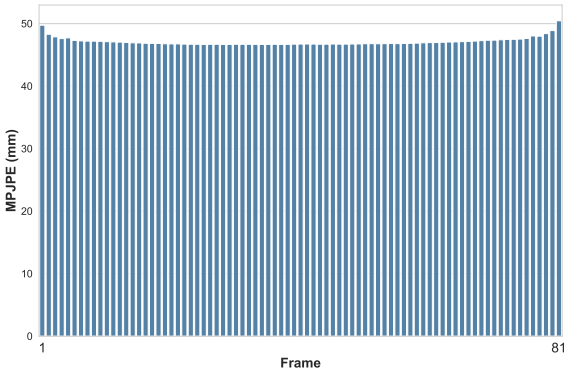

Figure 5.2: MPJPE in millimeters for the 3D pose of each input frame of TGraphNet. Note that middle frames are estimated with better accuracy due to larger temporal context from future and past frames.

Protocol #1	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	**Avg.** ↓
Pavlakos et al. [PZD18] CVPR2018	48.5	54.4	54.4	52.0	59.4	65.3	49.9	52.9	65.8	71.1	56.6	52.9	60.9	44.7	47.8	56.2
Pavllo et al. [Pav+19] ICCV2019(T=243)(†)	45.2	46.7	43.3	45.6	48.1	55.1	44.6	44.3	57.3	65.8	47.1	44.0	49.0	32.8	33.9	46.8
Zhao et al. [Zha+19b] CVPR2019	47.3	60.7	51.4	60.5	61.1	49.9	47.3	68.1	86.2	55.0	67.8	61.0	42.1	60.6	45.3	57.6
Cai et al [Cai+19a] CVPR2019(T=7)(†)	44.6	47.4	45.6	48.8	50.8	59.0	47.2	43.9	57.9	61.9	49.7	46.6	51.3	37.1	39.4	48.8
Xu et al. [XT21] CVPR2021	45.2	49.9	47.5	50.9	54.9	66.1	48.5	46.3	59.7	71.5	51.4	48.6	53.9	39.9	44.1	51.9
Chen et al. [Che+20] TCSVT2021(T=81)(†)	42.1	43.8	41.0	43.8	46.1	53.5	42.4	43.1	53.9	60.5	45.7	42.1	46.2	32.2	33.8	44.6
Zhao et al. [ZWT22] CVPR2022	45.2	50.8	48.0	50.0	54.9	65.0	48.2	47.1	60.2	70.0	51.6	48.7	54.1	39.7	43.1	51.8
Banik et al. [BGK21] ICIP2021	48.0	52.4	52.8	52.7	58.1	71.1	51.0	50.4	71.8	78.1	55.7	53.8	59.0	43.0	46.8	56.3
Zheng et al. [Zhe+21] ICCV2021(T=81)(†)	41.5	44.8	39.8	42.5	46.5	51.6	42.1	42.0	53.3	60.7	45.5	43.3	46.1	31.8	32.2	44.3
Zeng et al. [Zen+21] ICCV2021(†)	43.1	50.4	43.9	45.3	46.1	57.0	46.3	47.6	56.3	61.5	47.7	47.4	53.5	35.4	37.3	47.9
Zhang et al. [Zha+22] CVPR2022(T=243)(†)	**37.6**	**40.9**	**37.3**	**39.7**	**42.3**	**49.9**	**40.1**	**39.8**	**51.7**	**55.0**	**42.1**	**39.8**	**41.0**	**27.9**	**27.9**	**40.9**
TGraphNet(T=81)(†)	40.9	46.1	42.4	44.5	50.0	58.3	43.9	44.7	56.4	62.5	46.2	43.6	47.7	32.2	34.4	46.7
Protocol #2	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	**Avg.** ↓
Pavllo et al [Pav+19] CVPR2019(T=243)(†)	34.2	36.8	33.9	37.5	37.1	43.2	34.4	33.5	45.3	52.7	37.7	34.1	38.0	25.8	27.7	36.8
Zhao et al. [ZWT22] CVPR2022	41.6	45.3	46.7	46.8	51.3	56.5	43.6	43.7	56.1	62.0	50.9	46.0	50.1	39.2	45.0	48.3
Banik et al. [BGK21] ICIP2021	36.6	41.5	41.6	42.3	43.9	52.0	39.4	38.7	56.2	61.4	44.4	41.2	48.0	33.9	38.9	44.0
Zheng et al. [Zhe+21] ICCV2021(T=81)(†)	32.0	34.2	31.7	33.7	34.4	39.2	32.0	31.8	42.9	46.9	35.5	32.0	34.4	23.6	25.2	33.9
Zhang et al. [Zha+22] CVPR2022(T=243)(†)	**30.8**	**33.1**	**30.3**	**31.8**	**33.1**	**39.1**	**31.1**	**30.5**	**42.5**	**44.5**	**34.0**	**30.8**	**32.7**	**22.1**	**22.9**	**32.6**
TGraphNet(T=81)(†)	31.9	35.8	33.3	36.0	38.3	44.4	32.6	34.4	45.8	49.9	37.3	33.2	37.1	25.3	28.1	36.6
MPJVE	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	**Avg.** ↓
Pavllo et al [Pav+19] CVPR2019(T=243)(†)	3.0	3.1	2.2	3.4	2.3	2.7	2.7	3.1	2.1	2.9	2.3	2.4	3.7	3.1	2.8	2.8
Chen et al. [Che+20] TCSVT2021(T=243)(†)	2.7	2.8	2.0	3.1	2.0	2.4	2.4	2.8	1.8	2.4	2.0	2.1	3.4	2.7	2.4	2.5
Zheng et al. [Zhe+21] ICCV2021(T=81)(†)	3.2	3.4	2.6	3.6	2.6	3.0	2.9	3.2	2.6	3.3	2.7	2.7	3.8	3.2	2.9	3.1
Zhang et al. [Zha+22] CVPR2022(T=243)(†)	**2.5**	**2.7**	**1.9**	**2.8**	**1.9**	**2.2**	**2.3**	**2.6**	**1.6**	**2.2**	**1.9**	**2.0**	**3.1**	**2.6**	**2.2**	**2.3**
TGraphNet(T=81)(†)	2.6	**2.7**	2.2	3.0	2.1	2.6	2.4	2.8	2.1	2.5	2.1	2.2	3.3	2.7	2.5	2.5

Table 5.1: MPJPE in millimeters under Protocol #1 (P1) and Protocol #2 (P2), and MPJVE on Human3.6M dataset. The top table summarizes P1, the middle table summarizes P2, and the bottom table summarizes MPJVE. T denotes the number of input frames to the temporal models. † means that the model uses temporal relationships. CPN [Che+18] predicted 2D keypoints are used as input. Lower MPJPE and MPJVE mean better performance. The best and the second-best scores are highlighted in bold and underlined formats respectively.

Protocol #1	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	**Avg.** ↓
Pavllo et al. [Pav+19] ICCV2019(T=243)(†)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	37.2
Chen et al. [Che+20] TCSVT2021(T=243)(†)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	32.3
Zheng et al. [Zhe+21] ICCV2021(T=81)(†)	30.0	33.6	29.9	31.0	30.2	33.3	34.8	31.4	37.8	38.6	31.7	31.5	29.0	23.3	23.1	31.3
Zhang et al. [Zha+22] CVPR2022(T=243)(†)	**21.6**	**22.0**	**20.4**	**21.0**	**20.8**	**24.3**	**24.7**	**21.9**	**26.9**	**24.9**	**21.2**	**21.5**	**20.8**	**14.7**	**15.7**	**21.6**
TGraphNet(T=81)(†)	31.6	34.5	28.7	31.8	32.3	36.8	36.1	31.1	36.9	37.3	32.5	31.5	33.3	24.6	26.9	32.7

Table 5.2: MPJPE in millimeters under Protocol #1 (P1) between the estimated 3D pose and the ground truth 3D pose on the Human3.6M dataset using the **ground truth 2D pose** instead of CPN detections as input. T denotes the number of input frames used by the method. The best and second-best results are highlighted in bold and underlined respectively.

predictions can be made without modifying the model itself and with minimal performance penalties.

5.2 Root Relative 3D Human Pose Estimation

This section is dedicated to the experiments with the baseline TGraphNet for root relative pose estimation. TGraphNet (traj), which additionally outputs the global position of the root joint is evaluated and discussed separately. First, the quantitative results of the models are summarized with a comparison against other state of the art models. Then qualitative and quantitative error analysis is performed along with ablation studies.

5.2.1 Quantitative Evaluation

Results on Human3.6M Dataset

Table 5.1 shows the quantitative results of TGraphNet with 81 frames input and the sliding window inference strategy described in the previous section. CPN-detected 2D joints are used as input. Models that utilize temporal information are marked with the † symbol. The temporal window is also mentioned if the information is available. Single-frame based state

of the art models such as GraFormer [ZWT22] and PoseGraphNet [BGK21] are also included in the comparison to highlight the benefits of using temporal context for feature extraction. TGraphNet outperforms all single frame-based models. State of the art GCN and transformer-based method GraFormer [ZWT22] lags behind TGraphNet by roughly **11%** in P1. This demonstrates the importance of utilizing temporal relationships and motion priors in addition to spatial features for more accurate 3D HPE.

As can be seen in the table 5.1, TGraphNet lags behind the state of the art spatio-temporal transformer model MixSTE [Zha+22] by **5.8mm** (**14%**) in P1 Protocol and by **4mm** (**12%**) in P2. However, the difference between spatial and temporal transformer-based model PoseFormer [Zhe+21] is only **2.6mm** (**5%**). Additionally, TGraphNet utilizes more efficient architecture and predicts 3D pose for multiple frames instead of a single frame. Moreover, TGraphNet outperforms PoseFormer in terms of MPJVE by roughly **24%** highlighting the importance of motion supervision with dedicated loss functions. Compared to the best model MixSTE in terms of MPJVE, TGraphNet is a close second, lagging only by **0.2**. TGraphNet is the top 2nd performer on a large number of actions. In *Discussion* action, it shares first place with MixSTE with MPJVE of **2.7**. Hence, STGCN based spatio-temporal model performs on par with transformer-based methods in terms of MPJVE. Nevertheless, transformer based methods are superior in terms of localization accuracy measured by MPJPE. TgraphNet outperforms VideoPose3D by Pavllo et al. [Pav+19] while using fewer frames (T = 243 vs T = 81). VideoPose3D utilizes purely temporal feature extraction via TCN compared to TGraphNet, which also utilizes spatial relationships between joints through GCNs, giving it the edge. The same applies to MPJVE, where VideoPose3D lags behind TGraphNet by **0.3**. Compared to spatio-temporal graph-based models, TGraphNet falls back only behind [Che+20] by about **4%**. However, it is worth noting that [Che+20] uses a significantly heavier network with *45.3M* parameters as opposed to TGraphNet that has *3.8M* parameters (only **8%** of [Che+20]). Regarding MPJVE, both show the same average error of **2.5**.

To find out an estimate of an upper bound of TGraphNet's positional accuracy, it was trained using ground truth 2D inputs instead of CPN-detected inputs. The hindering effect of coming from the 2D detector is eliminated. Table 5.2 reports MPJPE per Protocol 1 after training TGraphNet using the same strategy described in section 5.1.3. The model performs roughly **30%** better than TGraphNet trained on CPN inputs. Hence, by using a more accurate 2D joint detector, the positional accuracy of TGraphNet can be increased by up to **14mm** (**30%**). Similar to results using CPN inputs, TGraphNet lags behind transformer-based methods. However, it is noteworthy that the difference between TGraphNet and [Che+20] is **0.4mm** with ground truth inputs, but the difference is more prominent (**2.1mm**) on CPN inputs. The same can be observed when comparing with VideoPose3D [Pav+19]. TGraphNet outperforms VideoPose3D by **4.5mm** when ground truth input is used, but the difference in performance is negligible with CPN inputs. This implies that [Che+20] and VideoPose3D [Pav+19] are more robust to noise coming from inaccuracies of the 2D detector. TGraphNet seems to be more sensitive to noise than other methods.

Generally, transformer-based methods such as PoseFormer and MixSTE [Zhe+21; Zha+22] perform the best when it comes to positional accuracy due to the ability to model global long-range relationships without pooling operations. However, this performance gain comes at the cost of computationally more complex architectures that are harder to train than GCN or TCN-based methods. Spatio-temporal graph convolutional methods such as TgraphNet and [Che+20] outperform purely temporal-based methods [Pav+19] and single frame models.

Generalization Performance

Following [Zhe+21], TGraphNet has been evaluated on the test set of the MPI-3DHP dataset to test the ability of the model to generalize to unseen data. Table 5.3 shows 3DPCK and

Methods	3DPCK ↑	AUC ↑
Mehta [Meh+17]	79.4	41.6
Pavllo et al. [Pav+19] (T=81)(†)	86	51.9
Pavllo et al. [Pav+19] (T=243)(†)	85.5	51.5
Chen et al. [Che+20] (T=243)(†)	<u>87.9</u>	<u>54</u>
Lin et al. [LL19] (T=25)(†)	83.6	51.4
Zheng et al. [Zhe+21] (T=81)(†)	**88.6**	**56.4**
TGraphNet (T=81)(†)	87.41	52

Table 5.3: 3DPCK and AUC on the MPI-3DHP test set. Models that use temporal context are marked with † and the number of input frames is stated. The best and the second-best scores are highlighted in bold and underlined. Higher score for both metrics means better.

Methods	f	FLOPs (M)	FPS	Parameters (M)	MPJPE ↓
Pavllo et al. [Pav+19] (T=81)	1	25.48	1121	12.79	47.7
Pavllo et al. [Pav+19] (T=243)	1	33.87	863	16.95	46.8
Chen et al. [Che+20] (T=81)	-	88.9	315	45.53	44.6
Chen et al. [Che+20] (T=243)	-	116	264	59.18	44.1
Zheng et al. [Zhe+21] (T=81)	1	1358	269	9.6	44.3
TGraphNet (T=81)	1	446	18	3.8	46.7
TGraphNet (T=81)	15	29.73	267	3.8	46.7
TGraphNet (T=81)	31	14.39	560	3.8	46.7
TGraphNet (T=81)	81	5.5	1451	3.8	47.0

Table 5.4: Comparison of computational complexity. FLOPs per frame, FPS, and the number of parameters alongside MPJPE are summarized in the table. f denotes the number of output 3D poses used after each run of the model. T denotes the number of input frames for each model. FPS of all models except TGraphNet were calculated on a single RTX 2080 GPU [Zhe+21]. The inference speed in the FPS of TGraphNet is calculated on an Intel Core I7-8750H laptop CPU.

AUC computed on the test set of the MPI-3DHP dataset. TGraphNet comes close to the best-performing models for generalization on the 3DHP dataset. In terms of 3DPCK, TGraphNet lags behind state of the art transformer based model, PoseFormer [Zhe+21], only by **1.2** although PoseFormer has slightly better AUC (**56.4 vs 52**). A similar situation arises when comparing to Chen et al. [Che+20]. TGraphNet has AUC, which is less by **2**. In terms of 3DPCK, the difference is almost negligible. Note that TGraphNet uses fewer frames as input (T = 81) and has fewer parameters. TGraphNet performs better than VideoPose3D [Pav+19], again highlighting better generalization ability due to the exploitation of both spatial and temporal contexts in the input through spatial-temporal graph convolutions.

Computational Cost Analysis

To assess the computational cost of the model, the estimated number of floating point operations (FLOPs) per output frame and frames per second (FPS) are reported in table 5.4. Alongside FLOPs, the number of parameters and MPJPE on Human3.6M per Protocol 1 are also reported in the table. Since TGraphNet follows the seq2seq paradigm, it is by default more efficient than a many-to-one model such as [Pav+19] as more than one 3D pose is estimated for each run of the model. As discussed in section 5.1.3, the middle 31 output frames of TGraphNet are combined with a sliding window strategy. Hence, the reported results also show FLOPs per frame and FPS with a different number of output frames considered to see the effect of output length on the performance. Even though the inference speed of TGraphNet is computed on a CPU (Intel Core i7-8750H), it outperforms all models when the whole output is used. Using the middle 31 frames yields 560FPS, which is faster than the transformer-based method PoseFormer [Zhe+21] and GCN based [Che+20]. This shows

the exceptional efficiency of the architecture of TGraphNet. It also has the least amount of parameters while maintaining comparable positional accuracy. When only one output pose is considered for 81 input frames, the efficiency drastically drops by **97%** compared to using 31 frames showing the efficient nature of the seq2seq paradigm. The same can be observed with FLOPs per output frame. When using only 1 output frame per each run, TGraphNet is not efficient. Moreover, positional accuracy stays the same, meaning there is no benefit in using only the middle frame per run. In summary, the spatio-temporal graph convolutional model is computationally more efficient than transformer-based methods. Comparable positional accuracy can be achieved with less demanding architecture and fewer parameters.

Error Analysis

Joint-Wise Error Analysis: To analyze the positional accuracy of TGraphNet for each joint individually, the average MPJPE aggregated by joint is plotted in figure 5.3. It is evident that joints such as *Knees*, *Feet*, *Elbows*, and *Wrists* are recovered with less accuracy. This can be attributed to the fact that these joints are more mobile, and their range of movements has more variance. Joints of *Torso* and *Hip* region are recovered with better accuracy since these joints are relatively more stable and have less movement freedom. This can be improved by utilizing a more accurate 2D joint location detector to minimize the noise in the input sequence [Sun+19].

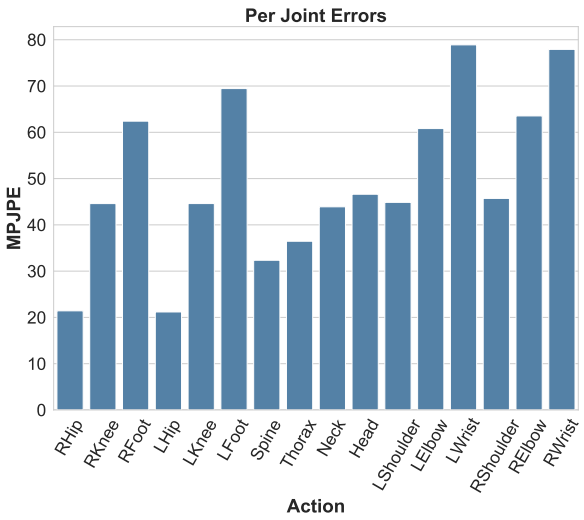

Figure 5.3: Mean per joint position error under Protocol 1 in millimeters for all joints calculated individually using all frames in the Human3.6M test set.

Error Distribution: Figure 5.4 shows the distribution of errors on the test set of Human3.6M.

The plot demonstrates a bimodal distribution where errors are concentrated in lower regions, but there is also a spike of extreme errors larger than 70mm of MPJPE. It can be noted, however, that most of the error lies within the range of 25mm to 50mm of MPJPE. Extreme errors can be attributed to specific actions where TGraphNet struggles to recover accurate 3D poses. The bias of the training set of the Human3.6M dataset towards actions that involve *Walking* and *Standing* [Meh+17] is one of the causes for extreme errors since models trained on Human3.6M only often struggle to generalize to in the wild scenarios. To alleviate this issue, training can be performed on multiple datasets to increase the variety of poses that the model can learn. Again, the usage of better 2D joint location detectors can also help.

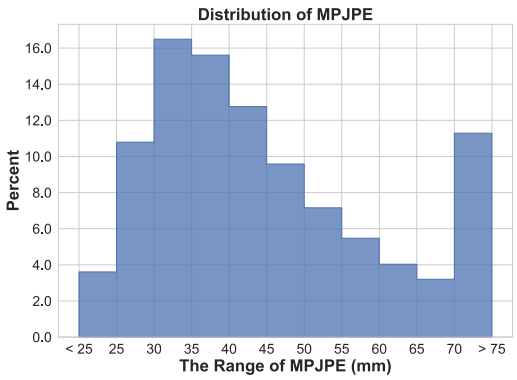

Figure 5.4: Histogram of MPJPE on Human3.6M test set for all actions. Errors are divided into small bins, and the percentage of errors in each bin is shown. Errors less than 25mm and greater than 70mm are shown at the extremes of the plot.

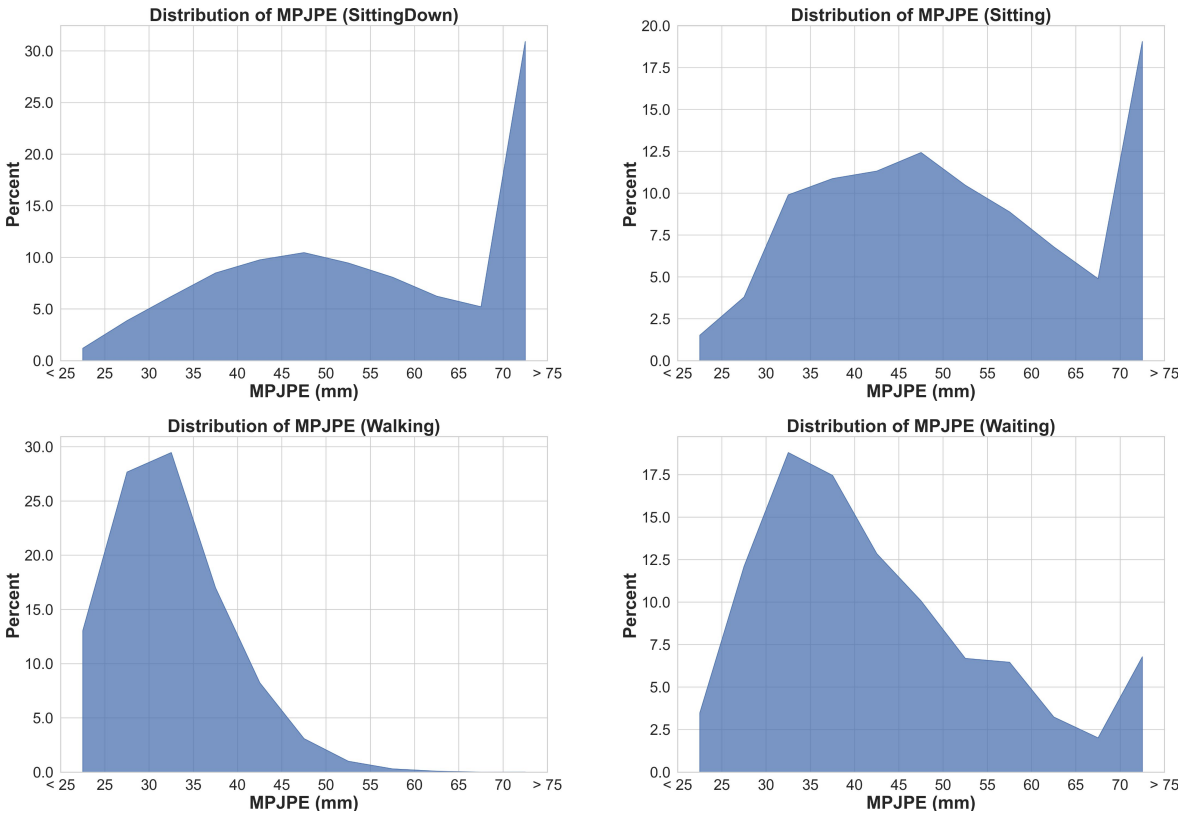

Figure 5.5: Distribution of MPJPE for 4 actions, Sitting, SittingDown, Walking, and Waiting. **Top**: Actions that have a large skew of error distribution towards higher values. **Bottom**: Actions for which reconstruction error is skewed towards lower values. In general, actions involving sitting poses are poorly reconstructed as opposed to actions involving standing positions.

Action-Wise Error Analysis: From table 5.1 of results on Human3.6M test set, *Sitting*, *SittingDown* and *Photo* are the top 3 hardest actions for TGraphNet to accurately reconstruct. The MPJPE error for *SittingDown* is **62.5mm**, which is roughly **38%** higher than the overall average MPJPE. MPJPE for *Sitting* (**58.9mm**) and *Photo* (**58.3mm**) are also significantly higher than the average. It is noteworthy that other methods [Zhe+21; Pav+19; Zha+22] also struggle the most with these actions, which shows a more global trend. The opposite can be said about actions such as *Walking* and *WalkingTogether*, which are reconstructed with less error. MPJPE for action *Walking* is **32.2**, which is **31%** less than the overall average. This trend can be observed with other methods as well.

Looking at the detailed distribution of error for actions *Sitting*, *SittingDown*, *Walking* and *Waiting* in figure 5.5, a skew of distribution is evident. Actions involving sitting have a skew towards the higher extreme where about **25-30%** of error is above **70mm**. The opposite is true for *Walking* and *Waiting*, where most of the error is concentrated in the lower extreme. Qualitative analysis with pose visualizations is performed to gain a full picture of what this error means for reconstructed 3D poses.

Figure 5.6: Model predictions versus ground truth 3D poses for 2 actions in Human3.6M, *Walking* and *Sitting*. A skeleton with green and orange bones represents ground truth predictions while pink and blue bones are used for the predicted pose. For each action, poses with low average and high reconstruction errors are shown.

5.2.2 Qualitative Analysis

Quantitative error measurements alone are insufficient for a complete model performance analysis. On the other hand, qualitative analysis with visualizations of reconstructed poses provides context on what the measured error means and how it relates to inaccuracies in predicted 3D poses. For instance, factors such as camera angle or occlusions can be visualized with qualitative analysis. To complete the evaluation of TGraphNet, this section presents qualitative samples and analysis on 3 datasets containing a variety of poses.

Qualitative Results on Human3.6M Dataset

Figure 5.6 shows the reconstructed 3D skeleton next to the ground truth skeleton for two actions *Walking* and *Sitting*. The ground truth skeleton is on the right of each plot. Various examples with low to high errors for both actions are displayed to visualize how prominent the inaccuracies are for each error district. The actions were chosen based on results in the table 5.1 since *Sitting* actions have the most significant amount of errors, whereas *Walking* actions have the least. For *Walking* action, TGraphNet consistently provides credible reconstructions of the 3D pose with discrepancies mostly around limbs (i.e., *Feet* and *Wrists*). In the pose with median error, *Left Wrist* is notably inaccurate compared to ground truth. There is also some inaccuracy in *Left Foot*. The rest of the pose is reconstructed potently. The worst-case scenario is an outlying result where legs are entirely inaccurate, caused by noisy CPN detections. As for the *Sitting* action, more significant errors are made by TGraphNet when the subject is shot from behind. This likely happens because of heavily occluded joints on arms and legs, which result in 2D detections of insufficient accuracy. In the worst-case scenario and 99th percentile error, legs and arms are reconstructed with significant inaccuracies. However, even though leg joints are occluded in the 95th and 75th percentile cases, the 3D pose is reconstructed with feasible accuracy.

Overall, larger errors in 3D pose prediction by TGraphNet happen when the person is engaged in a stationary action such as *Sitting*. Motion information in such actions is sparser when compared to *Walking* since the person is not moving as much. Hence, motion features are less helpful, and temporal models lose the ability to extract useful information from a sequence. This can be further buttressed by the fact that the gap in performance between single frame and temporal models is less in stationary actions such as *Sitting* or *Waiting*. For instance, the single frame based method proposed by Xu et al [XT21] lags behind VideoPose3D by **2.4mm** in *Sitting* action whereas in *Walk Together* action, where rich motion features can be extracted, the difference is **10.2mm**. In addition, occluded scenes are another major cause of inaccurate predictions. To cope with this issue, better 2D joint detectors or special occlusion robust training can be performed.

Qualitative Results on MPI-3DHP Dataset

Figure 5.7 visualizes the reconstructed and ground truth skeletons on the MPI-3DHP dataset. Actions with low to large reconstruction errors are demonstrated. Again, the trend of discrepancies in reconstructed limb joints can be noted in the figure. In the median and 75th percentile error plots, the error largely stems from incorrect positions of feet and wrists. In the 95th percentile error plot, there are discrepancies in the legs. In the case of the 99th percentile and the worst case, a difference in the scale of predicted and ground truth skeletons can be noticed. The average length of bones in the predicted skeleton is larger.

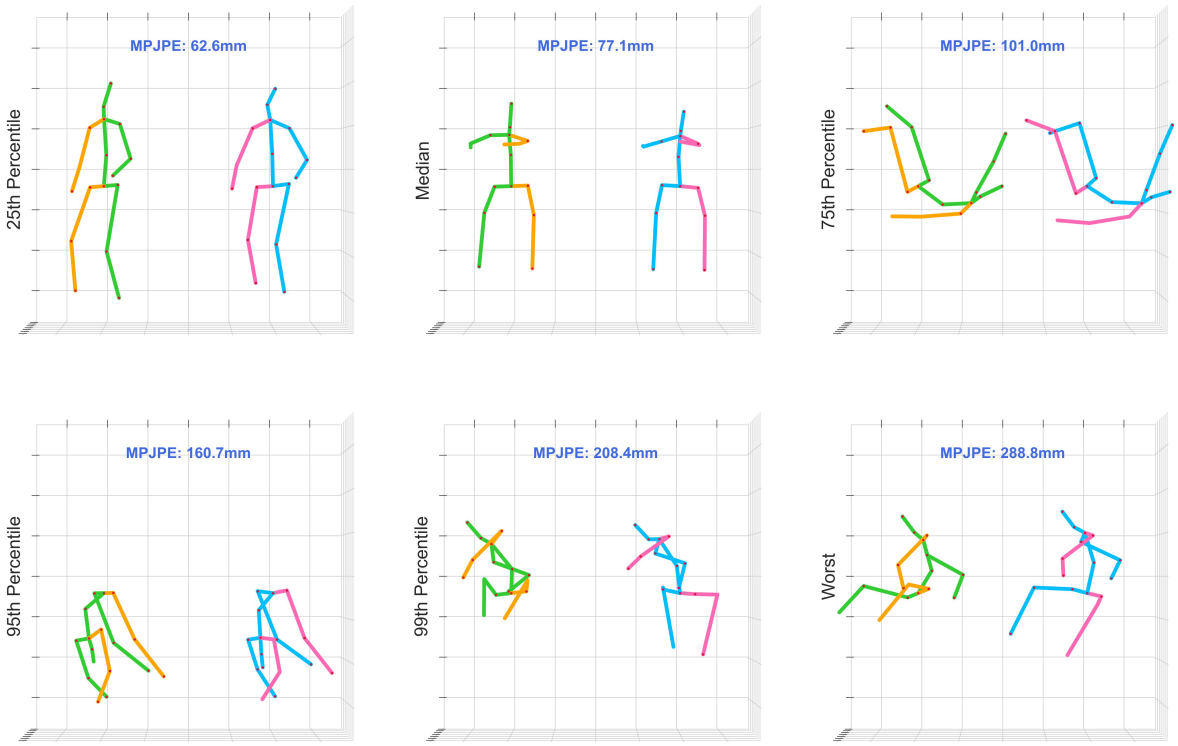

Figure 5.7: Model predictions versus ground truth 3D poses on MPI-3DHP. A skeleton with green and orange bones represents ground truth predictions while pink and blue bones are used for the predicted pose. For each action, poses with low average and high reconstruction errors are shown.

Qualitative Results on MPI-3DPW Dataset

Figure 5.8 shows the reconstructed 3D skeleton next to the ground truth skeleton for two actions *Fencing* and *Running for Bus* of the PW3D dataset. Looking at the results, a distinct type of error in reconstruction can be noticed. While the overall pose looks correct in most cases, the scale of the reconstructed skeleton or its overall size is smaller than the ground truth skeleton. The reason behind this discrepancy is the drastic difference in camera and filming setup between Human3.6M and MPI-3DPW datasets. Figure A.1 in Appendix visually compares a typical scene from Human3.6M with a scene from the PW3D dataset. The image in PW3D is wider, and the subject is further away from the camera because of unconstrained filming space. As a result, the input 2D pose is smaller in scale resulting in anomalous output 3D poses. Being trained on Human3.6M, TGraphNet has implicitly adapted to the scale of input and output skeletons of that particular filming setup. So, TGraphNet reconstructs 3D poses with the wrong scale. This shows a general limitation of 2D to 3D lifting methods since most are trained on 3D data in the camera space. So, 2D to 3D lifting methods are frequently constrained to a specific filming setup, and deviation from it can cause significant prediction errors. This issue can potentially be remedied via finetuning to target camera setups after pre-training models on large-scale datasets such as Human3.6M.

Figure 5.8: Model predictions versus ground truth 3D poses for 2 actions in MPI-3DPW, *Fencing* and *Running for Bus*. A skeleton with green and orange bones represents ground truth predictions while pink and blue bones are used for the predicted pose. For each action, poses with low average and high reconstruction errors are shown.

5.2.3 Ablation Study

Several ablation studies are conducted to analyze and verify the components of TGraphNet's architecture. First, architectural parameters and their effect are studied. Then, the effect of the length of the input sequence and a combination of loss functions is tested. Finally, the adaptive adjacency matrix and its contribution to the extraction of spatio-temporal relationships are analyzed.

Architecture Parameters

Various combinations of architectural parameters are tested to find the optimal TGraphNet architecture. The number of STGCN blocks, N_i, in the downsampling stage and hidden dimension h_i, are varied. These parameters are also shown in architecture figure 4.2. The results of this ablation are summarized in table 5.5. The architecture with the first hidden dimension $h_1 = 64$ and 2 STGCN blocks after the first layer leads to the result with the best MPJVE. However, changing h_1 to 128 leads to better reconstruction accuracy. Hence, the optimal parameters are chosen to match the ones in the 3rd row of the table.

A	N_1	N_2	N_3	N_4	h_1	h_2	h_3	MPJPE ↓	MPJVE ↓
Adaptive	1	1	1	1	64	128	256	48.2	2.6
Adaptive	1	2	2	2	64	128	256	47.4	**2.3**
Adaptive	1	2	2	2	128	128	256	**46.7**	2.5
Static	1	2	2	2	128	128	256	52.8	2.5

Table 5.5: Ablation study on different architecture parameters. The evaluation is performed on the Human3.6M test set under Protocol 1. MPJPE and MPJVE are reported for each combination of parameters. **Top** rows summarize the results with various parameters and learnable adjacency matrices. **Bottom** row shows the result with best-performing parameters but with static adjacency matrices. The best results are highlighted in bold.

Loss Functions

Table 5.6 summarizes the results of training TGraphNet with different combinations of losses. Firstly, training with no loss for motion supervision already achieves MPJVE (**2.6**), which is comparable with state of the art. With the addition of MPJVE and TCL Losses [Zha+22], MPJVE improves by **15%** decreasing to **2.2** from **2.6**. However, MPJPE becomes slightly worse. Training TGraphNet with motion loss [Wan+20] results in the best MPJPE error of **46.7**,

even though MPJVE lags behind *MPJPE + MPJVE + TCL Loss* combination. Supervision of the trajectory of each joint at multiple time scales using motion loss [Wan+20] improves the accuracy of the reconstructed 3D pose. Hence, *MPJPE + Motion Loss* combination is chosen for supervising TGraphNet.

Loss	MPJPE ↓	MPJVE ↓
MPJPE Loss	47.3	2.6
MPJPE Loss + MPJVE Loss + TCL Loss	47.4	**2.2**
MPJPE Loss + Motion Loss[Wan+20]	**46.7**	2.5

Table 5.6: Ablation study of the loss functions used to supervise TGraphNet. The evaluation is performed on the Human3.6M test set under Protocol 1. MPJPE and MPJVE are reported for each configuration. The best result is highlighted in bold.

Number of Input Frames

Model	MPJPE ↓	MPJVE ↓
TGraphNet(T = 1)	52.4	-
TGraphNet(T = 9)	50.3	3.8
TGraphNet(T = 27)	48.9	2.6
TGraphNet(T = 81)	**46.7**	**2.5**

Table 5.7: Ablation study on a different number of input frames to TGraphNet. The evaluation is performed on the Human3.6M test set under Protocol 1. MPJPE and MPJVE are reported for each configuration. The best result is highlighted in bold.

To test the effect of the length of the input sequence of 2D poses, an ablation is conducted where TGraphNet is trained and evaluated on the Human3.6M dataset with 4 different input

lengths, namely $T = \{1, 9, 27, 81\}$.

The results are summarized in table 5.7. All the models are trained according to the procedure described in section 5.1.3. The model with 81 input frames performs the best in terms of both MPJPE and MPJVE, highlighting the importance of large temporal context and including even a small temporal window of T = 9 results in an average **2.1mm** improvement in MPJPE compared to single frame model. Expanding the window to T = 27 yields another **1.4mm** improvement. The improvement in motion velocity is even more significant when going from 9 frame input to 27 frame input. There is an improvement of **1.2**, which is roughly **32%**. Finally, going from 27 frames to 81 frames improves MPJPE by **2.2mm** while the difference between MPJVE is almost negligible (**0.1**). This ablation shows that a large temporal window is beneficial for modeling realistic motions. Longer inputs improve the quality of reconstruction and motion, which is especially visible when moving from short (T = 9) to medium (T = 27) length inputs. Figure 5.9 visually demonstrates the benefits of utilizing multi-frame input with trajectory supervision. The trajectory of joints predicted by a single-frame model is choppy and contains jitter. The multi-frame model with low MPJVE, on the other hand, produces smooth motion.

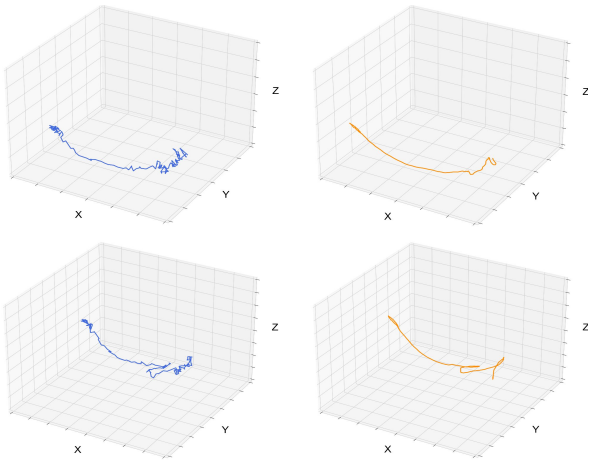

Figure 5.9: Trajectory of the left elbow and left wrist joints predicted by single-frame and multi-frame models. The blue line represents the trajectory predicted by the single-frame model. Orange is used to plot the trajectory predicted by the multi-frame model with 81 input frames. **Top**: the trajectory of the left elbow joint. **Bottom**: the trajectory of the left wrist joint.

Adaptive Adjacency Matrices

TGraphNet was trained and evaluated on Human3.6M with static adjacency matrices to test the effect of adaptive adjacency matrices. The bottom row of table 5.5 shows MPJPE and MPJVE achieved by TGraphNet with static adjacency matrices instead of adaptive. As can be seen in the table, fixing the adjacency matrices to reflect natural skeletal connections in the human body drastically drops the accuracy of TGraphNet from **46.7mm** MPJPE to **52.8mm** MPJPE (**13%**). Therefore, adaptive adjacency matrices are beneficial for capturing connectivity beyond natural connections in the human body resulting in better accuracy.

Additionally, Figures 5.10 visualize the learned spatio-temporal adjacency matrices of all neighborhood groups as heatmaps. Since STGCN processes the sequence in groups of three adjacent skeletons, each square block in the adjacency matrix represents links of 17 joints in that particular timestamp. The timestamp value is also shown next to joint names on horizontal and vertical axes for easier reading. The figures show that originally initialized links corresponding to skeletal connections in each neighborhood group (see section 4.3.1) still largely persist after training. Additionally, TGraphNet has captured links between joints that are not physically connected. It can also be observed that even though all temporal links were initialized to 0, TGraphNet has learned temporal connections through adaptive adjacency matrices. Each joint strongly associates with its temporal neighbors in the *Self* link matrix. Moreover, the link is stronger between joints in directly adjacent frames. For instance, links between joints in frames 1 and 2 are stronger than between frames 1 and 3. The strongest links, however, are spatial links within each frame, highlighting that TGraphNet tends to focus more on skeletal relations between body parts rather than temporal ones. TGraphNet has

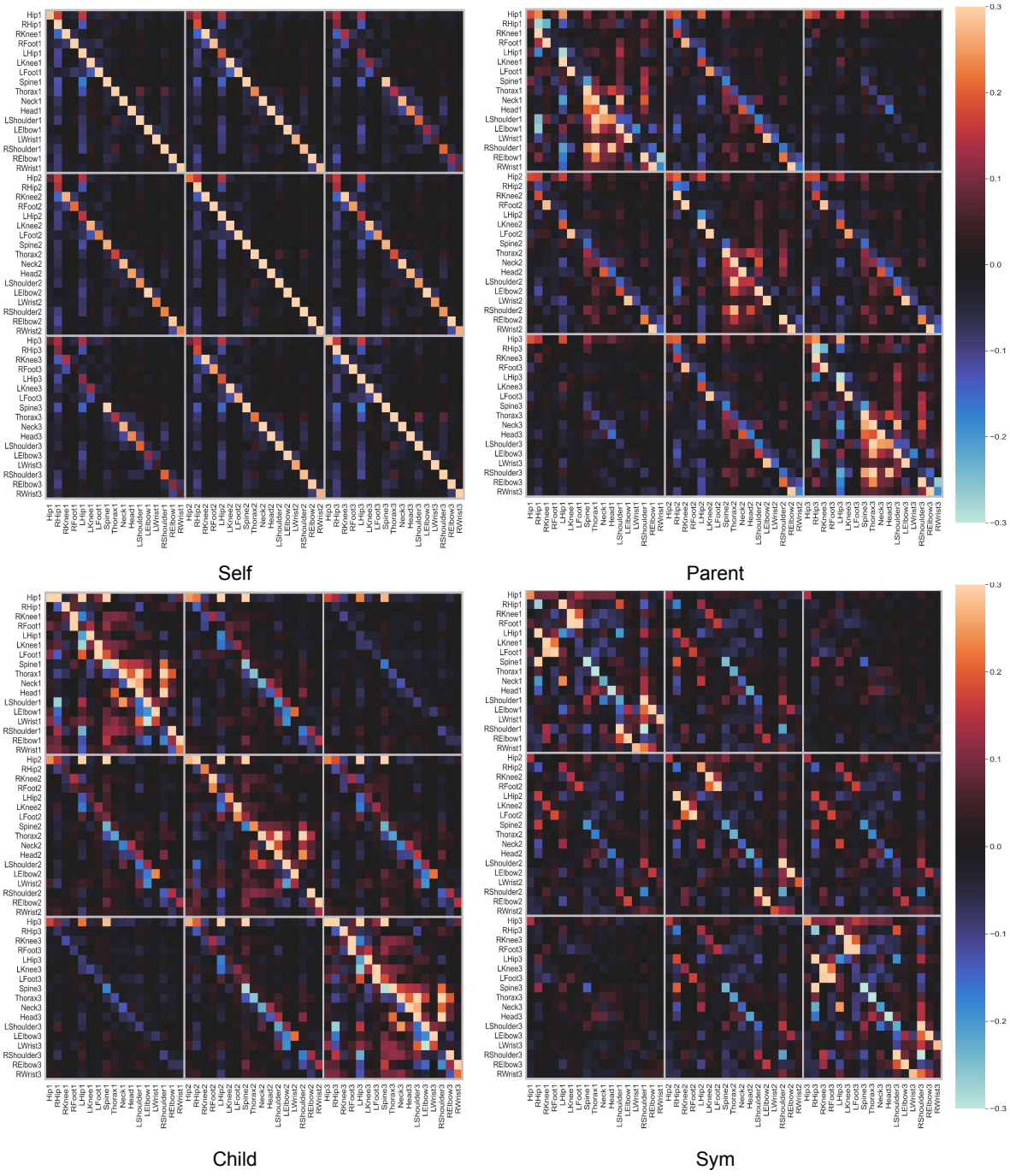

Figure 5.10: Adaptive adjacency matrices for all neighborhood groups learned by TGraphNet during training. Adjacency matrices contain temporal and spatial links. The name of the joint and its timestamp are mentioned on the horizontal and vertical axes.

also learned some links that connect different joints across time. In the matrix corresponding to *Self* link group, *Hip* joint is connected to *LHip* and *RHip* joints in adjacent frames. This highlights the ability of TGraphNet to learn associations of body parts across time depending on their semantic role in the body. In the *Symmetric* link group, TGraphNet captured relationships between symmetric body parts across time and each skeleton individually. Fixing adjacency matrices would prevent TGraphNet from learning meaningful spatial-temporal relationships shown in the figures leading to worse accuracy.

5.2.4 Summary

To summarize, the exploitation of temporal and spatial relationships with multi-scale feature extraction results in an improvement in terms of 3D pose reconstruction error. Gradually increasing the number of input frames shows that temporal context is essential for the model to estimate an accurate and realistic motion, as demonstrated in table 5.7. Additionally, supervision of motion with the motion loss [Wan+20] decreases reconstruction error further. However, the most beneficial is the inclusion of adaptive adjacency matrices allowing the model to capture spatial and temporal relations not restricted by initialized skeletal connections and making adjacency matrices adaptive results in roughly **12%** improvement in terms of MPJPE.

Combining adaptive adjacency matrices and spatial-temporal graph convolution results in an efficient sequence-to-sequence human pose estimation architecture. TGraphNet can potently reconstruct 3D poses from input most of the time. However, extensive qualitative and quantitative analyses demonstrate that there is still room for improvement. One major hindrance to accurate pose estimation is the scenes with occluded body parts (see figure 5.6), which lead to noisier input provided by 2D joint detectors. Evaluation of TGraphNet on ground truth data shows the approximate upper bound of performance that can be achieved with inputs of better quality (see table 5.2). Another limitation is the implicit adaptation of the model to the camera system used to capture the training data. As shown in figures 5.7 and 5.8, this implicit adaptation to the training camera system leads to inconsistencies in reconstructed 3D poses. The most notable inconsistency is the size of the estimated 3D skeleton. Finetuning or post-processing can possibly resolve this issue, but the fact remains that models trained in camera-constrained settings are not readily generalizable to arbitrary settings. These issues leave the door open for further research to improve 3D pose estimation models.

5.3 Trajectory Estimation

The estimation of the 3D position of the root joint is a more challenging task than root relative joint location estimation because more depth cues are needed to ensure accurate localization of the Hip joint in the 3D space in the input. In contrast, root relative 3D pose estimation assumes the skeleton is centered at the origin. Thus, the task reduces to accurate localization of joints with respect to the Hip joint, which is solved with impressive accuracy by utilizing skeletal symmetries in the 2D input.

This section presents a trajectory model named *TGraphNet (traj)*, which aims to address the absence of depth cues by leveraging the motion characteristics extracted from the sequence of 2D skeletons. Specifically, the model utilizes the motion of the skeleton alongside its change of scale in the sequence. The variations in scale serve as implicit depth cues, where a smaller skeleton indicates that the person is moving away from the camera, while a larger skeleton suggests that the person is closer to the camera. Numerous experiments have been conducted to validate the effectiveness of utilizing motion features to estimate the global 3D pose.

The trajectory model is mainly identical to TGraphNet in terms of architecture. It additionally includes a regression TCN that predicts the position of the *Hip* joint in the 3D camera space (see figure 4.2). The rest of the joints are still predicted relative to the root. So, to obtain the global position of each joint, the coordinates of the predicted root joint are added to root-relative coordinates. Thus, the reconstructed sequence of 3D poses now has a trajectory, and the skeleton is no longer centered at the origin of the coordinate system. The trajectory

Protocol #1	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	**Avg.↓**	**Hip Avg. ↓**	**Hip X↓**	**Hip Y↓**	**Hip Z↓**
TGraphNet(T=81)	40.9	46.1	42.4	44.5	50.0	58.3	43.9	44.7	56.4	62.5	46.2	43.6	47.7	32.2	34.4	46.7	-	-	-	-
TGraphNet (traj) (T=81)(⋆)	42.6	48.1	46.8	46.2	51.0	58.5	45.7	45.4	61.4	66.3	48.2	44.8	49.1	34.4	36.9	48.6	-	-	-	-
TGraphNet (traj) (T=81)(#)	47.2	54.9	52.0	52.8	56.6	65.7	51.3	50.5	66.7	77.4	54.3	51.7	54.6	39.7	42.4	54.9	106.8	17.9	18	98.6
MPJVE	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	**Avg.↓**	**Hip Avg.↓**	**Hip X↓**	**Hip Y↓**	**Hip Z↓**
TGraphNet(T=81)	2.6	2.7	2.2	3.0	2.1	2.6	2.4	2.8	2.1	2.5	2.1	2.2	3.3	2.7	2.5	2.5	-	-	-	-
TGraphNet (traj) (T=81)(⋆)	2.8	2.9	2.2	3.2	2.3	2.6	2.5	3.0	2.3	2.7	2.3	2.4	3.5	3.0	2.7	2.6	-	-	-	
TGraphNet (traj) (T=81)(#)	2.9	3.1	2.4	3.4	2.4	2.8	2.7	3.1	2.4	2.9	2.4	2.5	3.7	3.2	2.9	2.8	4.7	-	-	-

Table 5.8: Comparison of TGraphNet and TGraphNet (traj) in MPJPE under Protocol #1 (P1) and MPJVE on Human3.6M dataset. The top table summarizes P1 and the bottom table summarized MPJVE. T denotes the number of input frames to the temporal model. ⋆ symbol means that only root relative 3D pose is considered for evaluation. # indicates that the position of the **Hip** joint is also factored in when computing MPJPE and MPJVE. The error of global pose localization in each axis is also reported. CPN [Che+18] predicted 2D keypoints are used as input. Lower MPJPE and MPJVE values indicate better performance.

model is additionally supervised with reprojection loss which computes the average error between the 2D input and the projection of reconstructed 3D into the image space. This way, the input and output poses are forced to be consistent with each other. After training the trajectory model in a similar regime as TGraphNet, a series of evaluations and ablations are performed in the sections below.

5.3.1 Quantitative Evaluation

Results on Human3.6M

Firstly, it can be noted that training for root joint coordinates results in approximately **4%** (2mm) deterioration in positional accuracy compared to regular TGraphNet. This can mean that sharing the same backbone feature extractor for global and root relative position estimation tasks hinders the model's performance. This is also in line with the finding in VideoPose3D [Pav+19], where the authors trained two separate networks for each task. The most drop in performance happens in the *Sitting* and *SittingDown* actions. This can be explained by the fact that subjects move significantly less in those actions making it harder for the motion model to extract temporal features. Including the coordinates of the Hip joint in evaluation increases the MPJPE by **13%** (6.3mm). The average error of the Hip joint alone is **106.8mm** which is significantly larger compared to the errors in the reconstruction of the rest of the joints (see figure 5.3). Moreover, most of the error is along the Z axis (**98.6**), which is expected since the input lacks depth cues. The overall quality of the motion measured by MPJVE also increases by **0.1** when trained for root joint positions. It further increases by **0.2** when the root joint is considered during evaluation. The average MPJVE of the global trajectory of the skeleton is **4.7**, which is **67%** larger than the overall average. These factors facilitate the idea that estimating the global position alongside the relative root pose of the skeleton is a complicated task. Lacking depth cues in the 2D joint locations significantly hinders the accuracy of the estimated depth of the skeleton.

Generalization Performance

To assess how well TGraphNet (traj) generalizes to unseen data, it was evaluated on the MPI-3DHP dataset. 3DPCK and AUC are reported for root relative pose only. The MPJPE for the global pose is reported separately. MPJPE for the global pose is additionally decomposed over each axis. Again, the performance slightly drops when including the prediction of the global pose. 3DPCK is reduced by roughly **4%** and AUC is dropped by **1**. The average MPJPE for the Hip joint is about **1186mm**, which is more than 10 times larger compared to the error on the Human3.6M test set (**106.8**).

Methods	3DPCK↑	AUC↑	Avg. Hip↓	Hip X↓	Hip Y↓	Hip Z↓
TGraphNet (T=81)	87.41	52	-	-	-	-
TGraphNet (traj) (T=81)	82.6	51	1186	95.3	28.3	1177

Table 5.9: 3DPCK and AUC on MPI-3DHP test set for TGraphNet and TGraphNet (traj). MPJPE for the root joint along its decomposition across axes is reported. ↑ indicates higher value is better and ↓ indicates that lower value is better.

The predicted location of the skeleton is roughly **1** meter away from its actual location. This means that the trajectory model cannot generalize to the MPI-3DHP dataset. The average error across the Z axis is **1177**, which encompasses most of the error. Therefore, the network struggles to estimate the depth of the skeleton more than on the Human3.6M test set. This means that the network learns to associate motion and scale features to depth on Human3.6M but fails on MPI-3DHP since the camera setup is significantly different.

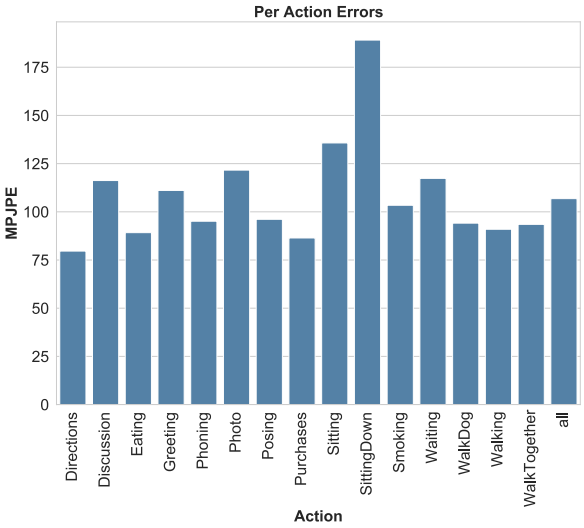

Figure 5.11: Average MPJPE of the **Hip** joint in millimeters for all actions calculated individually using all frames in the Human3.6M test set.

Error Analysis

Figure 5.11 plots the average positional error of the predicted root joint for each action in the Human3.6M test set separately. As can be observed in the figure, the largest error is made in *Sitting* and *SittingDown* actions. Errors for these two actions exceeded **125mm** mark with *SittingDown* having above **175mm** reconstruction error between ground truth and predicted location of the skeleton. This can be explained by the lack of motion in these actions. Moreover, the Human3.6M dataset has a significant skew towards actions involving standing or walking [Meh+17], which is another major cause of large errors in sitting poses. The errors for *Walking*, *WalkTogether*, and *WalkDog* are all below the threshold of **100mm**. Figure 5.12 visualizes the distribution of errors for 4 actions, *Sitting*, *SittingDown*, *Walking*, and *WalkTogether* in more detail. Regarding walking actions, a large portion of error is roughly concentrated in **25-125mm** region. Actions involving sitting have a large spike of errors concentrated in the region above **250mm**. Particularly, *SittingDown* action has the most noticeable skew towards large errors. The following section presents the results of qualitative analysis to analyze the effect of significant errors on the reconstructed poses.

5.3.2 Qualitative Evaluation

Qualitative Analysis on Human3.6M Dataset

Plots of ground truth and predicted 3D poses with the global position are visualized in figure 5.13. The predicted skeleton in black is overlayed on the ground truth skeleton to visualize the relative differences between the two. Two actions, *Walking* and *SittingDown*, are used for visualization as one has a low reconstruction error. In contrast, the other has the highest

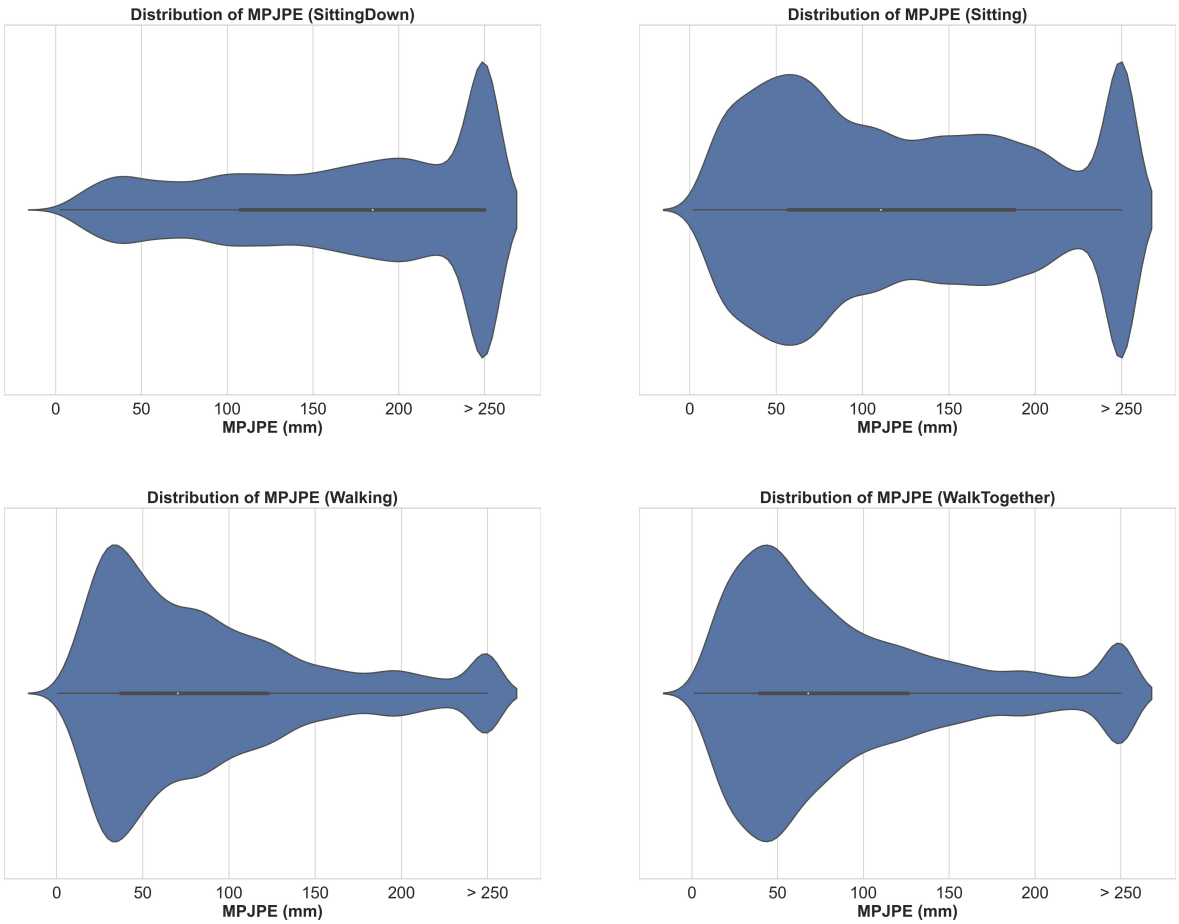

Figure 5.12: Distribution of MPJPE of the global position for 4 actions, Sitting, SittingDown, Walking, and WalkTogether. **Top**: Actions that have a large skew of error distribution towards higher values. **Bottom**: Actions for which reconstruction error is skewed towards lesser values. Actions involving sitting poses in general are poorly reconstructed when compared to actions involving standing positions.

error among all actions (see figure 5.11). The plot shows various poses with low and high errors for both actions. Visually, the reconstructed poses for *Walking* actions look potently close to the ground truth up to the pose with 95th percentile error. The predicted skeleton is located near the ground truth skeleton with a slight offset. In the 99th percentile and worst-case scenarios, the predicted skeleton is ahead of the actual skeleton highlighting the challenge of depth estimation from a set of sparse 2D detections.

Plots for *SittingDown* action also demonstrate the consequences of incorrect depth estimation for the global position. In the worst-case scenario, the scale of the predicted skeleton is considerably smaller than the scale of the actual skeleton. This means the model predicted the skeleton to be further away from the camera. Moreover, the legs' positions are likely inaccurate due to occlusions since the scene was shot from behind. The 95th percentile plot shows that the Hip joint was localized with a large horizontal and vertical offset. The same can be noticed in the median error plot where the predicted skeleton is to the right of its ground truth counterpart.

Qualitative Analysis on MPI-3DHP and 3DPW Datasets

Figure 5.14 and 5.15 demonstrate plots of predicted and ground truth skeletons on MPI-3DHP and 3DPW datasets overlayed atop each other. Essentially the same situation as in Human3.6M plots can be observed. The predicted location of the skeleton is at an offset

Figure 5.13: Model predictions versus ground truth 3D poses for 2 actions in Human3.6M, *Walking* and *Sitting-Down*. **Red** skeleton represents the ground truth skeleton while **Black** is the predicted skeleton. Ground truth and predicted skeletons are overlayed on each other to visualize relative differences in the global pose of **Hip** joint and other joints. MPJPE error including the global position is mentioned in the top of each plot.

majority of the time, and the estimated depth of the skeleton is largely incorrect. In the worst-case plot of the 3DPW dataset, the predicted 3D skeleton is significantly further from the

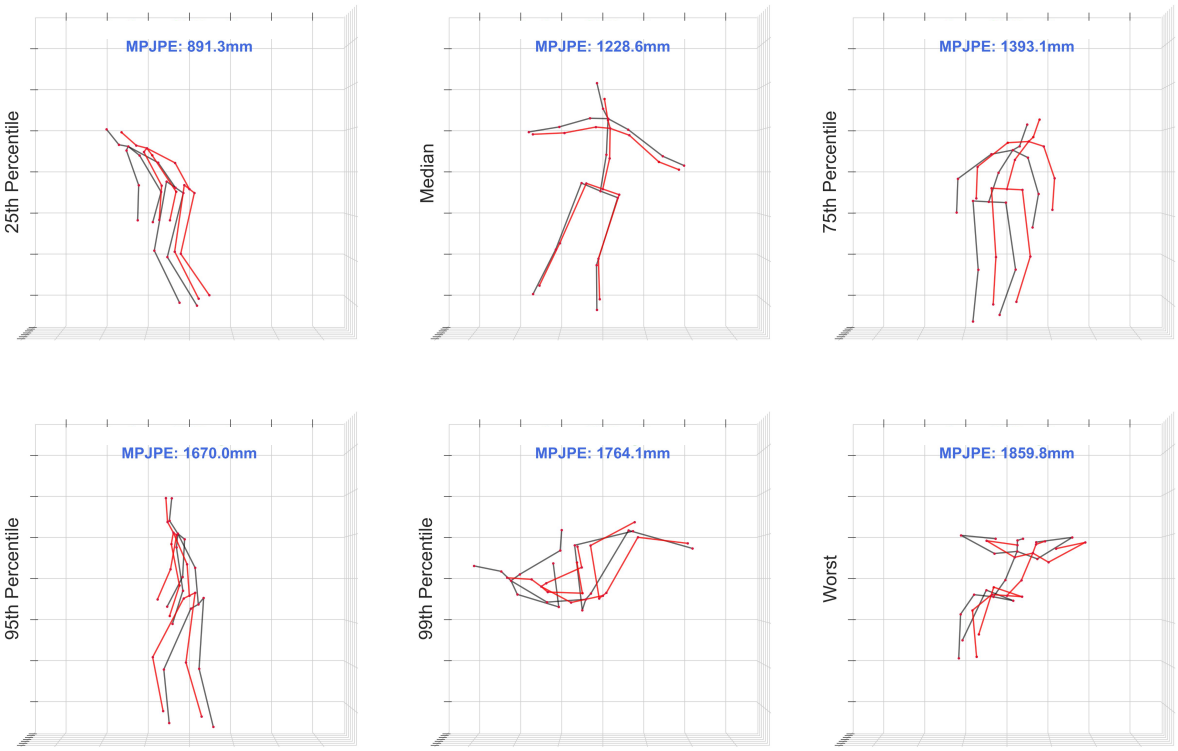

Figure 5.14: Model predictions versus ground truth 3D poses on MPI-3DHP. **Red** skeleton represents the ground truth skeleton while **Black** is the predicted skeleton. Ground truth and predicted skeletons are overlayed on top of each other to visualize the relative difference in the global pose of **Hip** joint and the rest of the joints.

camera than the ground truth skeleton. This is the case for the 99th percentile plot as well. This shows that the model adapts to the training dataset's camera setup and cannot generalize to other camera characteristics. That is to say, the model learns to find correspondence between the scale of the skeleton and body parts in 2D space and depth in 3D space, but only in the camera space of the training dataset. Hence, the average depth error is nearly 10 times larger on MPI-3DHP than on Human3.6M.

Coupled with quantitative results, it is evident that the trajectory model is not able to infer depth using motion information accurately. This is especially true for camera setups significantly different from the training setup. This situation also happens in root relative pose estimation, as shown in the previous section (see section 5.2.2). Therefore, incorporating additional information from other inputs, such as images, is necessary to improve the positional accuracy of both models.

5.3.3 Ablation Study

Several ablation studies are conducted to analyze and verify the components of TGraphNet (traj) architecture. First, architectural parameters and their effect are studied. Then, the effect of the length of the input sequence and a combination of loss functions is tested. The effect of the aforementioned architectural choices is evaluated on the global pose estimation accuracy.

Architecture Parameters

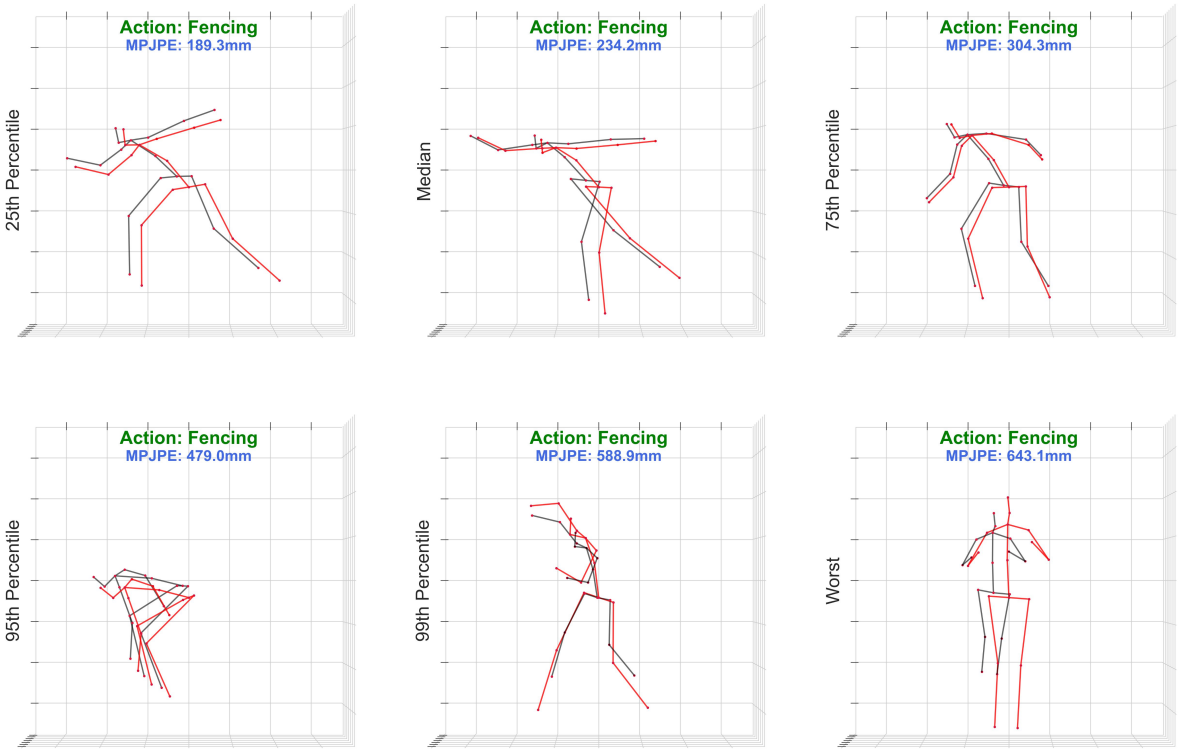

Figure 5.15: Model predictions versus ground truth 3D poses of *Fencing* action MPI-3DPW. **Red** skeleton represents the ground truth skeleton while **Black** is the predicted skeleton. Ground truth and predicted skeletons are overlayed on top of each other to visualize the relative difference in the global pose of **Hip** joint and the rest of the joints.

Different architectural parameters are tested to determine which combination results in the best performance for the TGraphNet (traj) model. The number of STGCN blocks, N_i, and hidden dimension h_i in the downsampling stage are varied (see 4.2). The results of the ablation study are summarized in table 5.10. The architecture with the first hidden dimension $h_1 = 128$ and 2 STGCN blocks after the first layer leads to the best MPJPE and MPJVE for both the relative root pose and the global pose.

This shows that performing both tasks is demanding, and a more expressive network is needed to increase the model's performance. More parameters significantly improve the localization performance of the Hip joint.

N_1	N_2	N_3	N_4	h_1	h_2	h_3	MPJPE ↓	MPJPE (Hip)↓	MPJVE↓	MPJVE (Hip) ↓
1	1	1	1	64	128	256	51.4	139.2	2.9	3.9
1	2	2	2	64	128	256	52.3	124.1	**2.6**	3.3
1	2	2	2	128	128	256	**48.6**	**106.8**	**2.6**	**3**

Table 5.10: Ablation study on different architecture parameters. The evaluation is performed on the Human3.6M test set under Protocol 1. MPJPE and MPJVE are reported for each combination of parameters. Metrics for the root **Hip** joint are computed separately and reported as well.

Loss	MPJPE↓	MPJPE (Hip)↓	MPJVE ↓	MPJVE (Hip)↓
MPJPE Loss	48.1	121.6	2.6	3.6
MPJPE Loss + Proj. Loss	49	143.2	2.7	3.1
MPJPE Loss + Motion Loss	**47.9**	112.3	2.7	3.6
MPJPE Loss + Motion Loss + Proj. Loss	48.6	**106.8**	**2.6**	**3**

Table 5.11: Ablation study on loss functions. The evaluation is performed on the Human3.6M test set. Metrics for the root joint are reported separately. The best result is highlighted in bold.

Loss Functions

Table 5.11 summarizes the results of training TGraphNet (traj) with different combinations of loss functions. Inclusion of reprojection loss improves MPJVE of the root joint by **14%** over training only with MPJPE loss. This shows that enforcing consistency between input and out-

put pose sequences is beneficial for accurate motion estimation. However, the localization accuracy suffers. On the other hand, the inclusion of Motion Loss [Wan+20] improves the localization performance leading to the best root relative MPJPE. Finally, combining reprojection and motion losses leads to the best metrics for global pose estimation.

Number of Input Frames

TGraphNet (traj) was trained with different input sequence lengths to determine the influence of temporal context and receptive field for the global pose estimation. The results are reported in table 5.12. The input sequence length has the most considerable effect on the velocity error of the whole skeleton and individual joints. Going from 9 to 27 frames improves MPJVE of the Hip joint by almost **40%**. MPJVE of all joints also improves roughly by the same amount. MPJVE of the root joint further improves by **42%** when increasing the input sequence length to 81. The localization accuracy of joints also increases with respect to the sequence length. This ablation shows the importance of the temporal context for full global motion estimation. Without a wide receptive field, smooth motion estimation becomes complicated for the model.

Model	MPJPE ↓	MPJPE (Hip) ↓	MPJVE ↓	MPJVE (Hip) ↓
TGraphNet(T = 1)	54	126.4	-	-
TGraphNet(T = 9)	51	115.8	5.1	8.5
TGraphNet(T = 27)	50	124.9	3.2	5.2
TGraphNet(T = 81)	**48.6**	**106.8**	**2.6**	**3**

Table 5.12: Ablation study on a different number of input frames to TGraphNet (traj). The evaluation is performed on the Human3.6M test set under Protocol 1. MPJPE and MPJVE are reported for each configuration. Metrics for the root joint are reported separately. The best result is highlighted in bold.

5.3.4 Summary

To summarize, training TGraphNet (traj) to predict the global pose and motion of the skeleton alongside root relative pose slightly drops the performance of the model compared to TGraphNet, which produces only root relative pose. This indicates that sharing the same architecture for both tasks harms the model's performance. Decoupled training of separate models for two tasks may be worthwhile to explore. Evaluation of the error between predicted and ground truth root (Hip) joint location Human3.6M dataset demonstrates that TGraphNet (traj) struggles to estimate the depth from the input 2D sequence correctly. The most considerable error occurs in the Sitting and SittingDown actions, which can be attributed to the subjects' limited movement in those actions, making it challenging for the motion model to extract temporal features (see figures 5.12 and 5.13). Thus, relying solely on the scale of the skeleton, its body parts, and their temporal changes within the 2D pose sequence is insufficient for accurately determining depth. To cope with this, additional depth cues extracted from images can be used in a multi-modal training fashion to enrich input 2D pose features.

Further quantitative and qualitative evaluation on other datasets, MPI-3DHP and MPI-3DPW, demonstrates that TGraphNet (traj) adapts to the camera system used for the Human3.6M dataset, which it was trained on. In other words, the model becomes excessively tailored to the specific camera perspectives and characteristics of the Human3.6M dataset, enabling it to establish associations between input 2D poses and their corresponding 3D representations within those particular camera setups. Consequently, TGraphNet (traj) struggles to adapt to variations in camera parameters, such as viewpoint, focal length, or imaging characteristics, which directly impact depth estimation accuracy. One way to counter this issue is to perform domain adaptation by finetuning the model after pretraining it on a large dataset like Human3.6M. This will theoretically enable the model to adapt to the target camera setup

while retaining its feature extraction capacity learned on the Human3.6M dataset. This idea remains to be explored in future work.

Chapter 6

Conclusion and Future Work

6.1 Conclusion and Discussion

This thesis focused on developing an efficient method for reconstructing a motion of a single person in 3D space by exploiting spatial-temporal relationships in videos. Two neural networks based on spatio-temporal graph convolutions, TGraphNet and TGraphNet (traj), were developed for estimating the 3D pose of persons from input 2D joint locations obtained from videos. TGraphNet is the basis for TGraphNet (traj), which, in addition to root-relative pose, also estimates the global position of the person in 3D space. Both models use graph convolutional networks and temporal convolutions to extract short-term spatial-temporal features and combine them for long-term global feature extraction. With the implementation of non-uniform graph convolution using learnable adjacency matrices, models capture skeletal features and relationships between joints in the human body without being restricted by fixed connections defined by the skeletal structure. Additionally, models are supervised with loss functions that enforce continuity on the output sequence of poses leading them to produce smooth and realistic motion.

TGraphNet, trained to reproduce root relative 3D human pose sequences, demonstrates performance close to state of the art at the same time being much more efficient and lightweight. Being a sequence-to-sequence model with motion supervision, it demonstrates the ability to reconstruct smooth motion with low-velocity error comparable to state of the art. Conducted studies show that TGraphNet gains performance with the increase in the receptive field, meaning that it effectively captures the temporal context in the input sequences. Moreover, making the adjacency matrix adaptive allows TGraphNet to extract richer connectivity features again, resulting in increased localization accuracy. While showing promising generalization performance, TGraphNet suffers from over-adaptation to the camera setup used in the training dataset. As a result, the model performs poorly when the input is captured on camera with significantly different characteristics, such as image plane size or focal length.

TGraphNet (traj) enriches the predicted root relative poses by additionally estimating the global position of the 3D skeleton. The main challenge of estimating the trajectory or global position of the skeleton in 3D space is the lack of depth information in a set of 2D joint locations. TGraphNet (traj) uses implicit depth cues in the input sequence of 2D poses to cope with this issue. Specifically, the size and scale of body parts in the 2D pose and their change across time serve as a rough indication of distance between the person and the camera because closer objects appear larger in the image plane. Evaluation and analysis of TGraphNet (traj) show that depth cues in motion are sparse for complete depth estimation. While the results may be feasible on the camera setup used in the training procedure, TGraphNet (traj) fails to estimate the depth of the skeleton in 3D space when used in a different camera setup provided in datasets other than Human3.6M. This again highlights the limitation of models

trained on specific camera setups, which is a general limitation not specific to TGraphNet and TGraphNet (traj).

To summarize, exploiting multi-level spatial-temporal relationships through the graph and temporal convolutions is an efficient approach for 3D human pose sequence estimation from videos. While root relative joint location estimation is generally performed with accuracy close to state of the art, global trajectory prediction still needs to be improved due to severely limited input features and over-adaptation to training camera setups.

6.2 Future Work

The evaluation of models proposed in this thesis highlights several worthwhile directions for improving the results.

Firstly, because models trained on the Human3.6M dataset suffer from over-adaptation to the camera system used in that dataset, it is worth exploring domain adaptation and fine-tuning methods. Pre-training tasks can be defined by training 2D to 3D lifting methods on large datasets and specific camera setups where models learn general-purpose features and representations. Then, the model can be fine-tuned to adapt the learned features to the target domain with different camera setups, thereby increasing the performance on specific tasks. Another potential solution is the inclusion of camera parameters in the training loop so the model has information about image size and focal length. Then, the model can be trained on multiple datasets with different camera setups where the model can learn to associate 2D pose to 3D poses being explicitly conditioned on the camera.

One of the main challenges in global pose estimation of the person in 3D space is the lack of depth information in 2D joint detections. Thus, the input 2D joint locations can be further enriched by depth features obtained from images via some neural network. Multi-modal settings can help models leverage depth information for more accurate generalization.

Another limitation specific to graph convolutional networks is the reliance on connectivity information independent of the input. In this case, the adjacency matrix captures global relationships between joints regardless of the input 2D pose, whether the person is sitting or standing. In contrast, transformers extract affinity information through self-attention, which is conditioned on the input, meaning that it can adapt to various actions that the person is performing. A possible solution to this limitation would be to design a graph convolution operation that also conditionally considers input features to modify the global adjacency matrix.

Lastly, one of the weaknesses of 2D to 3D lifting methods is their dependence on accurate 2D detections. When a 2D detector fails and provides noisy input, the resulting 3D pose will inevitably be less accurate. As observed when evaluating TGraphNet, the performance can be significantly improved with more accurate 2D input. Thus, better and more efficient 2D detectors are necessary for accurate 2D to 3D lifting methods.

Appendix A

PW3D and Human3.6M Scene Comparison

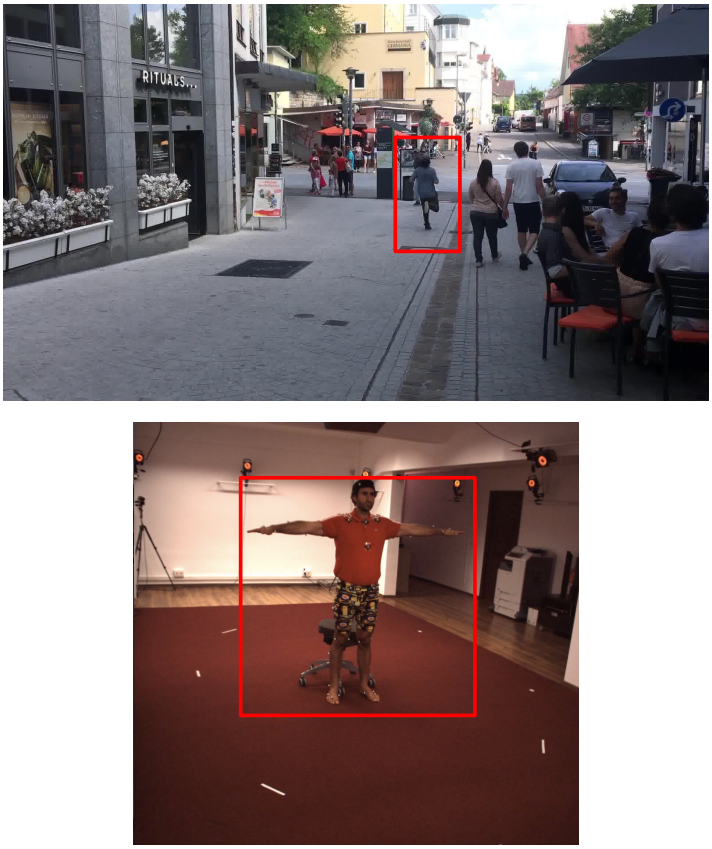

Figure A.1: Top shows a scene from *Run For Bus* action from PW3D dataset. **Bottom** shows a typical scene from Human3.6M dataset. The subject is surrounded in a red bounding box. The aspect ratio of the images are preserved.

Figure A.1 shows a comparison between a typical Human3.6M scene and a scene from Run for Bus action from PW3D dataset. The image in PW3D dataset on top is much wider than a typical scene from Human3.6M dataset. Moreover, the subject distances himself further away from the camera. As a result, the subject and its 2D skeleton are smaller resulting in anomalous predictions by the model. Due to these differences, the model is not able to generalize.
A typical camera in Human3.6M dataset has $f = [1145, 1143]$ focal length with $h = w = 1000$ resolution. PW3D utilizes cameras with focal length $f = [1969, 1961]$ and resolution $h = 1920$, $w = 1080$.

Bibliography

[BGK21] Banik, S., GarcÍa, A. M., and Knoll, A. “3D Human Pose Regression Using Graph Convolutional Network”. In: *IEEE International Conference on Image Processing (ICIP)*. ISSN: 2381-8549. Sept. 2021, pp. 924–928. DOI: 10.1109/ICIP42928.2021.9506736.

[Ban+23] Banik, S., Gschoßmann, P., Garcia, A. M., and Knoll, A. *Occlusion Robust 3D Human Pose Estimation with StridedPoseGraphFormer and Data Augmentation*. 2023. arXiv: 2304.12069 [cs.CV].

[Bru+14] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. “Spectral Networks and Locally Connected Networks on Graphs”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Bengio, Y. and LeCun, Y. 2014. URL: http://arxiv.org/abs/1312.6203.

[Cai+19a] Cai, Y., Ge, L., Liu, J., Cai, J., Cham, T.-J., Yuan, J., and Thalmann, N. M. “Exploiting Spatial-Temporal Relationships for 3D Pose Estimation via Graph Convolutional Networks”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, Oct. 2019, pp. 2272–2281. ISBN: 9781728148038. DOI: 10.1109/ICCV.2019.00236. URL: https://ieeexplore.ieee.org/document/9009459/.

[Cai+19b] Cai, Y., Ge, L., Liu, J., Cai, J., Cham, T.-J., Yuan, J., and Thalmann, N. M. “Exploiting Spatial-temporal Relationships for 3D Pose Estimation via Graph Convolutional Networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[Che+20] Chen, T., Fang, C., Shen, X., Zhu, Y., Chen, Z., and Luo, J. “Anatomy-aware 3D Human Pose Estimation in Videos”. In: *arXiv preprint arXiv:2002.10322* (2020).

[Che+18] Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., and Sun, J. “Cascaded pyramid network for multi-person pose estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7103–7112.

[CTH20] Chen, Y., Tian, Y., and He, M. “Monocular human pose estimation: A survey of deep learning-based methods”. In: *Computer Vision and Image Understanding* 192 (2020), p. 102897. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2019.102897. URL: https://www.sciencedirect.com/science/article/pii/S1077314219301778.

[DBV16] Defferrard, M., Bresson, X., and Vandergheynst, P. “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”. In: *Advances in Neural Information Processing Systems*. Ed. by Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. Vol. 29. Curran Associates, Inc., 2016.

[Dos+21] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[Duv+15] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. "Convolutional Networks on Graphs for Learning Molecular Fingerprints". In: *Advances in Neural Information Processing Systems*. Ed. by Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf.

[Gil+17] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. "Neural Message Passing for Quantum Chemistry". In: ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1263–1272.

[Has+22] Hassanin, M., Khamis, A., Bennamoun, M., Boussaid, F., and Radwan, I. "Crossformer: Cross Spatio-Temporal Transformer for 3D Human Pose Estimation". en. In: *SSRN Electronic Journal* (2022). ISSN: 1556-5068. DOI: 10.2139/ssrn.4213439. URL: https://www.ssrn.com/abstract=4213439.

[He+16] He, K., Zhang, X., Ren, S., and Sun, J. "Deep Residual Learning for Image Recognition". In: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '16. Las Vegas, NV, USA: IEEE, June 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: http://ieeexplore.ieee.org/document/7780459.

[IS15] Ioffe, S. and Szegedy, C. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Bach, F. and Blei, D. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 448–456. URL: https://proceedings.mlr.press/v37/ioffe15.html.

[Ion+14] Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2014).

[Kea+16] Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. "Molecular graph convolutions: moving beyond fingerprints". en. In: *Journal of Computer-Aided Molecular Design* 30.8 (Aug. 2016), pp. 595–608. ISSN: 0920-654X, 1573-4951. DOI: 10.1007/s10822-016-9938-8. URL: http://link.springer.com/10.1007/s10822-016-9938-8.

[KW17] Kipf, T. N. and Welling, M. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations (ICLR)*. 2017.

[Li+23] Li, W., Liu, H., Ding, R., Liu, M., Wang, P., and Yang, W. "Exploiting Temporal Contexts With Strided Transformer for 3D Human Pose Estimation". In: *IEEE Transactions on Multimedia* 25 (2023), pp. 1282–1293. DOI: 10.1109/TMM.2022.3141231.

[LL19] Lin, J. and Lee, G. H. "Trajectory Space Factorization for Deep Video-Based 3D Human Pose Estimation". In: *BMVC*. 2019.

[Liu+20] Liu, K., Ding, R., Zou, Z., Wang, L., and Tang, W. "A comprehensive study of weight sharing in graph networks for 3D human pose estimation". In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 318–334.

[Lop+15] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6 (Oct. 2015), 248:1–248:16.

[Mar+18] Marcard, T. von, Henschel, R., Black, M., Rosenhahn, B., and Pons-Moll, G. "Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera". In: *European Conference on Computer Vision (ECCV)*. Sept. 2018.

[Mar+17] Martinez, J., Hossain, R., Romero, J., and Little, J. J. "A simple yet effective baseline for 3D human pose estimation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

[Meh+17] Mehta, D., Rhodin, H., Casas, D., Fua, P., Sotnychenko, O., Xu, W., and Theobalt, C. "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.

[PZD18] Pavlakos, G., Zhou, X., and Daniilidis, K. "Ordinal Depth Supervision for 3D Human Pose Estimation". In: *Computer Vision and Pattern Recognition (CVPR)*. 2018.

[Pav+19] Pavllo, D., Zürich, E., Feichtenhofer, C., Grangier, D., Brain, G., and Auli, M. "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.

[RKK18] Reddi, S. J., Kale, S., and Kumar, S. "On the Convergence of Adam and Beyond". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=ryQu7f-RZ.

[Sri+14] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[Sun+19] Sun, K., Xiao, B., Liu, D., and Wang, J. "Deep high-resolution representation learning for human pose estimation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5693–5703.

[TV98] Trucco, E. and Verri, A. *Introductory techniques for 3-D computer vision*. Upper Saddle River, NJ: Prentice Hall, 1998. ISBN: 9780132611084.

[Vas+17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[Wan+21] Wang, J., Tan, S., Zhen, X., Xu, S., Zheng, F., He, Z., and Shao, L. "Deep 3D human pose estimation: A review". In: *Computer Vision and Image Understanding* 210 (2021), p. 103225. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2021.103225. URL: https://www.sciencedirect.com/science/article/pii/S1077314221000692.

[Wan+20] Wang, J., Yan, S., Xiong, Y., and Lin, D. *Motion Guided 3D Pose Estimation from Videos*. 2020. arXiv: 2004.13985 [cs.CV].

[Wu+21] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (Jan. 2021), pp. 4–24. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2020.2978386. URL: https://ieeexplore.ieee.org/document/9046288/.

[XT21] Xu, T. and Takano, W. "Graph stacked hourglass networks for 3D human pose estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16105–16114.

[Zen+21] Zeng, A., Sun, X., Yang, L., Zhao, N., Liu, M., and Xu, Q. "Learning Skeletal Graph Neural Networks for Hard 3D Pose Estimation". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 11416–11425.

[Zha+22] Zhang, J., Tu, Z., Yang, J., Chen, Y., and Yuan, J. "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 (CVPR)*. June 2022, pp. 13232–13242.

[Zha+19a] Zhang, S., Tong, H., Xu, J., and Maciejewski, R. "Graph convolutional networks: a comprehensive review". en. In: *Computational Social Networks* 6.1 (Dec. 2019), p. 11. ISSN: 2197-4314. DOI: 10.1186/s40649-019-0069-y. URL: https://computationalsocialnetworks.springeropen.com/articles/10.1186/s40649-019-0069-y.

[Zha+19b] Zhao, L., Peng, X., Tian, Y., Kapadia, M., and Metaxas, D. N. "Semantic Graph Convolutional Networks for 3D Human Pose Regression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[ZWT22] Zhao, W., Wang, W., and Tian, Y. "GraFormer: Graph-oriented Transformer for 3D Pose Estimation". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA: IEEE, June 2022, pp. 20406–20415. ISBN: 9781665469463. DOI: 10.1109/CVPR52688.2022.01979. URL: https://ieeexplore.ieee.org/document/9880048/.

[Zhe+22] Zheng, C., Wu, W., Chen, C., Yang, T., Zhu, S., Shen, J., Kehtarnavaz, N., and Shah, M. *Deep Learning-Based Human Pose Estimation: A Survey*. 2022. arXiv: 2012.13392 [cs.CV].

[Zhe+21] Zheng, C., Zhu, S., Mendieta, M., Yang, T., Chen, C., and Ding, Z. "3D Human Pose Estimation With Spatial and Temporal Transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 11656–11665.

[Zho+20] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. "Graph neural networks: A review of methods and applications". en. In: *AI Open* 1 (2020), pp. 57–81. ISSN: 26666510. DOI: 10.1016/j.aiopen.2021.01.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S2666651021000012.