A Simple Neural Network Approach for 3D Human Mesh Keypoints Estimation with a Novel Dataset[*]

Valerio Spagnoli[1], Flavio Volpi[1]

[1] *Sapienza University of Rome*

Abstract

This project presents a novel approach to extracting keypoints from 3D human meshes for pose estimation, with a focus on developing a new dataset and a simple yet effective neural network architecture. Given the lack of suitable datasets for 3D human pose estimation based on mesh structures, we created the Human Mesh Keypoints Extraction dataset by deriving it from the Human Body Segmentation dataset. The proposed neural network leverages mesh edge features to predict 12 keypoints, representing essential body joints, such as the neck, shoulders, elbows, wrists, hips, knees, and ankles. A key challenge in 3D keypoint extraction is the unordered nature of keypoints across meshes, which we addressed using Hungarian Matching in the loss function to optimally match predicted and ground truth keypoints. Our method was evaluated using metrics such as Mean Per Joint Position Error (MPJPE) and Percentage of Correct Keypoints (PCK), demonstrating good accuracy and robust performance. Despite the limited size of the dataset, the results show promising potential for the broader application of our model in 3D human pose estimation.

Keywords

3D Deep Learning, 3D Human Pose Estimation, Keypoints Extraction, Mesh Edge Features.

1. Highlights

- **Creation of a New Dataset**: Developed the Human Mesh Keypoints Extraction dataset from the Human Body Segmentation dataset, addressing the lack of suitable datasets for 3D human pose estimation using meshes.
- **Simple Neural Network Architecture**: Proposed a small and efficient neural network that leverages mesh edge features to predict 12 keypoints representing critical human body joints.
- **Keypoint Association Solution**: Overcame the challenge of unordered keypoints using Hungarian Matching in the loss function, ensuring optimal matching between predicted and ground truth keypoints.
- **Promising Results**: Achieved good accuracy in keypoint prediction, as evaluated by metrics such as Mean Per Joint Position Error (MPJPE) and Percentage of Correct Keypoints (PCK).
- **Scalability and Future Work**: Demonstrated the model's potential for broader applications, with future improvements focusing on expanding the dataset and refining the architecture for more complex human poses.

2. Introduction

Keypoint extraction from 3D human meshes is a crucial challenge in applications such as motion capture, human pose estimation, and virtual avatars. Keypoint extraction task aims to accurately localize a set of predefined anatomical points on a person. In the context of 3D human meshes, precisely identifying key joint positions is fundamental for effectively reconstructing human movement and understanding body posture.

Traditionally, this task for 2D images attracted many researchers in the field of computer vision, trying to solve problems such as human action recognition [1], pose tracking [2] and motion prediction [3]. The usage of deep learning methods greatly simplified the feature extraction directly from the images, achieving also excellent performances in all these challenges. Another step forward was done designing approaches for multi-persons scenes [4], allowing to efficiently ascribe and divide keypoints across all the persons in the image.

In the last years, deep learning models have been used widespread also for 3D data analysis, covering all representation types a three-dimensional space can have [5, 6, 7, 8, 9]. On this wave of 3D data analysis, some frameworks to deal with non-Euclidean spaces [10] and specifically with meshes [11] are born.

About the specific task of 3D human keypoints extraction, it has become a popular task, especially in the 3D pose estimation starting from images. Neural networks permit the recovery of hidden information of an object, from a two-dimensional space to three-dimensional representation of it, and many approaches have been tried [12, 13, 14]. However, the results are not as satisfactory as in the 2D case, due to depth ambiguities and mainly to the lack of datasets.

Instead, the extraction of the keypoints from meshes has been diffused for many years, due to their large diffusion in the 3D animations field, which include for instance videogames and movie animation. Specifically, these application areas must solve the mesh skeleton extraction challenge, i.e. to retrieve the skeleton of the mesh with the puprose to animate the entire mesh animating just its skeleton. Until now, this problem has been solved designing deterministic algorithms [15, 16, 17]

In this work, we propose a novel approach to keypoint extraction from 3D human meshes by using a neural network to predict key joint locations. Specifically, we focus on extracting 12 critical keypoints that represent essential joints of the human body: the neck, shoulders, elbows, wrists, hips, knees, and ankles. Two illustrative examples are shown in Figure 1.

The input features of the model are inspired by [11], creating an input tensor that can be treated as an image-like tensor, to which normal 2D convolution can be applied. As explained before, the lack of 3D datasets is a large problem

[*] You can use this document as the template for preparing your publication. We recommend using the latest version of the ceurart style.
[*] Corresponding author.
[†] These authors contributed equally.

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

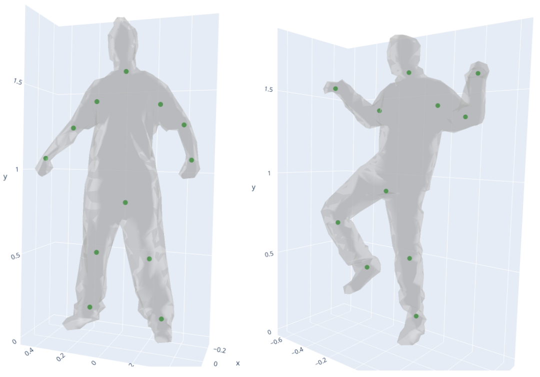

Figure 1: Examples of keypoints. The keypoints include the neck, shoulders, elbows, wrists, hips, knees, and ankles.

in the field of 3D deep learning. We could not find a suitable dataset for human pose estimation on meshes; therefore, we derived our dataset from the Human Body Segmentation dataset [18], consisting of 370 training models from SCAPE [19], FAUST [20], MIT [21], and Adobe Fuse [22], along with 18 test models from the SHREC07 [23] (human) dataset. The meshes in this dataset are segmented into eight labels, following the annotations in *Learning 3D Mesh Segmentation and Labeling* [24].

This paper is structured as follows: we begin in Section 3 with an overview of traditional keypoint extraction techniques from 2D images, transitioning into 3D cases. This section also summarizes various deep learning approaches for deriving 3D points from 2D images, as well as algorithms designed to extract skeletal structures directly from 3D meshes.

In Section 4, we introduce the Human Mesh Keypoints Extraction dataset, derived from the Human Body Segmentation dataset, which addresses the lack of suitable datasets for 3D human pose estimation on meshes. We also present the key algorithm used to create this dataset and discuss the challenges related to symmetrical keypoints.

Section 5 details the proposed neural network architecture, which leverages mesh edge features to predict keypoints. It includes an explanation of the edge features used, the neural network structure, and the innovative use of Hungarian Matching in the loss function to address the unordered keypoint association problem.

Finally, in section 6 we describe the training process, preprocessing steps, and the metrics used to evaluate the model's performance. We also present both analytic and visual results, highlighting the effectiveness of the model in predicting keypoints across various poses.

3. Related Works

Human keypoints estimation is a fundamental problem, mostly diffused in pose estimation task. We will resume the main approaches have been developed to solve it for both 2D and 3D scenarios.

2D human pose estimation from images. Predominatly, the architectures divide in two main groups, according to the followed approach: top-down or bottom-up. **Top-down** frameworks usually consist of two primary components: an object detector responsible for generating human bounding boxes and a pose estimator for identifying human keypoint locations. The object detector plays a critical role in the effectiveness of human proposal detection and, consequently, impacts the accuracy of pose estimation. Conversely, the pose estimator serves as the central element of the framework, directly influencing the precision of the pose estimation process. Thus, the top-down framework is highly adaptable and can be continuously enhanced with advancements in both object detection and pose estimation technologies. Different paradigms have been applied: regression-based methods [25, 26, 27] guarantee high efficiency in terms of computational time, but they neglect the spatial context of the corresponding body part. Thus heatmap-based methods were developed [28, 29, 30, 31] to achieve higher accuracy on joint location, thanks to a probabilistic approach instead of localizing the keypoints by determined coordinates. The cons of these approaches is the computational overhead. **Bottom-up** methods mainly perform directly the keypoints estimation, ensuring lighter computations. They can be divided in human center regression-based [32], associate embedding-based [33] and part field-based [4].

3D human pose estimation from images. Even if methods which use temporal information (monocular images or multi-view images at different times) are available, we will focus just on non-temporal approaches. The approaches can be splitted in two macro groups, according to if one monocular image is used or multi-view images are used. In the former we can divide the approaches that tries a direct keypoints estimation [34], those ones that takes advantage of the 2D pose estimation methods to lift 2D information in 3D pose [35, 14], and the SMPL model based ones [36]. On the other side, keypoints estimation from multi-view images can be summarized in just one class, where the methods differ on the different fusion strategies of multi-view information, such as using epipolar geometry [37] or putting images from different points-of-view inside the same CNN model [38].

3D Skeleton extraction from meshes. Three categories of algorithm regarding skeleton extraction exist. The former is the class of medial axis transform approaches, which provide region-based shape features: usually, Voronoi diagram is used to construct the skeleton [39]. The second one is the generalized potential field methods, where a distance field is constructed to extract the skeleton, and voxel-based methods are frequently used [40]. Finally, decomposition-based methods use Reeb graph to generate a one-dimensional skeleton [15]. The Reeb graph concept employs a continuous function, typically a height function, to represent the topological structure and capture topological changes, such as merging or splitting. By applying this continuous function, the target object is divided into multiple sections, and connecting the central positions of these sections generates the primary skeleton [41].

3D deep learning frameworks. Geometric deep learning [10] is a branch which attempts to extend artificial neural networks to non-Euclidean domains. Pytorch3D [42] is an extension of Pytorch [43] which provides operators for 3D deep learning. MeshCNN [11] is a framework which allows to deal with 3D meshes analogously to classical Convolutional Neural Networks.

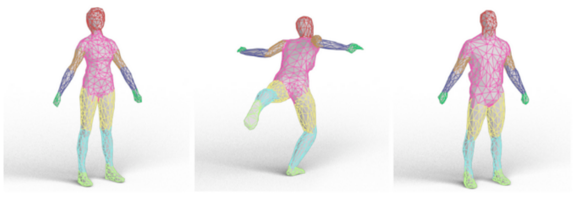

Figure 2: Examples from the Human Body Segmentation dataset.

4. Implementation

4.1. Human Mesh Keypoints Extraction Dataset

The Human Body Segmentation dataset comprises 370 training samples and 18 test samples. The meshes are segmented into 8 classes: head, torso, forearms, hands, arms, thighs, shinbones, and feet. Three illustrative examples are presented in Figure 2.

To construct our dataset, we base our approach on the following idea: since the 8 regions represent key parts of the human body, the connections between these parts correspond to the joints of the human body, thereby defining the keypoints of the skeleton.

The labels in the Human Body Segmentation dataset are assigned to the edges; for example, edge number 142 of the mesh belongs to class 1 (head). This implies that each vertex can belong to one or two classes, as it may be shared by two or more edges that correspond to different classes.

4.1.1. Algorithm to derive the new dataset

Using this concept, we implemented the following algorithm to identify the keypoints of a mesh:

1. Utilize the segmentation labels to determine the class of each edge and construct a list of tuples, `edges`, where the first element is the edge (v_1, v_2) and the second element is the corresponding class.
2. From the list `edges`, identify the classes to which each vertex belongs, creating a dictionary, `vertices`, where the key is the vertex v and the value is a set of classes c_1, c_2 associated with that vertex. Then, consider only the vertices belonging to two different classes, resulting in a dictionary, `useful_vertices`. An illustrative example of useful vertices is shown in Figure 3a.
3. Using the dictionary `useful_vertices`, remove from the list `edges` all edges whose vertices belong to the same class. This results in a new list of edges, `useful_edges`, containing only those edges connecting vertices from different classes.
4. Treating the list `useful_edges` as a graph, construct the `adjacent_list` for each vertex in this list. This results in a dictionary where $key = v_n$ and $value = [v_i, v_j, \ldots, v_k]$ with $i, j, \ldots, k \neq n$ for each entry.
5. Using the `adjacent_list`, identify the *connected components* of the graph `useful_vertices`. An illustrative example of connected components is presented in Figure 3b. Each connected component contains all vertices that belong to the same two distinct classes, c_1 and c_2. Figure 3c shows the vertices belonging to the same connected component.
6. Determine the keypoints of the mesh as the mean of the vertices within each connected component. An illustrative example of keypoints is presented in Figure 3d.

4.1.2. Human Body Segmentation to Human Mesh Keypoints Extraction

The number of *connected components*, and consequently the number of keypoints, is not consistent across all meshes. Specifically, from the training split of the Human Body Segmentation dataset, we derived the following distribution:

- 3 meshes with 14 connected components;
- 62 meshes with 13 connected components;
- 253 meshes with 12 connected components;
- 54 meshes with 11 connected components;
- 9 meshes with 10 connected components;

Similarly, from the test split, we derived:

- 1 meshes with 13 connected components;
- 8 meshes with 12 connected components;
- 6 meshes with 11 connected components;
- 2 mesh with 10 connected components;
- 1 mesh with 8 connected components;

After a visual analysis, we observed that the connection between the torso and the thighs was not consistent across all meshes. Specifically, in some meshes, this connection formed a single connected component, while in others, it resulted in two distinct connected components. An illustrative example of this difference is shown in Figure 4.

This inconsistency affected the total number of keypoints. To avoid discarding a significant number of meshes, we decided to unify the connection between the torso and thighs into a single class across all meshes. After applying this adjustment, the distribution of connected components changed as follows:

- 33 meshes with 13 connected components;
- 285 meshes with 12 connected components;
- 54 meshes with 11 connected components;
- 9 meshes with 10 connected components;

Similarly, for the test split, we derived the following distribution:

- 7 meshes with 12 connected components;
- 8 meshes with 11 connected components;
- 1 mesh with 10 connected components;
- 1 mesh with 9 connected components;
- 1 mesh with 8 connected components.

Finally, the majority of the meshes consist of 12 connected components (i.e. 12 keypoints), so we retained only these meshes for further analysis. Given the relatively small number of meshes with 12 keypoints in the test set, we combined **292 meshes** (285 from the training split and 7 from the test split) into a single annotated set, each with **12 keypoints**.

The remaining 107 meshes, which do not include any annotations, were stored in a separate set for visual testing purposes.

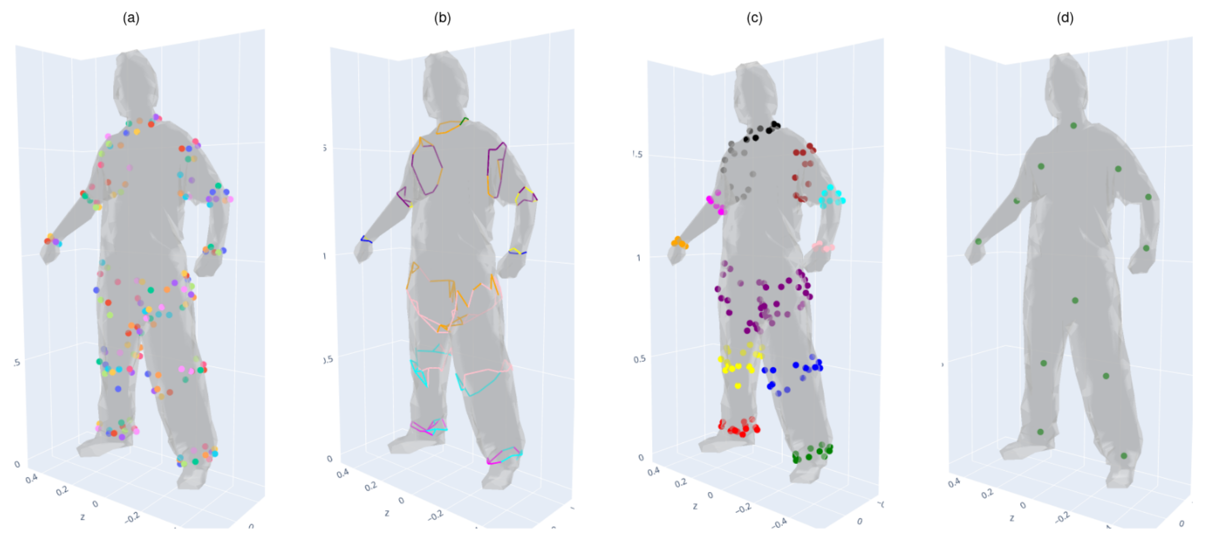

Figure 3: Illustrative examples derived from the various steps of the algorithm used to create the dataset are presented. In sub-figure *a*, the useful vertices of the mesh are shown, representing the vertices that belong to two distinct segmentation classes. In sub-figure *b*, the connected components of the useful edges are displayed, i.e., the edges that have at least one useful vertex. Sub-figure *c* highlights the vertices within the connected components. Finally, in sub-figure *d*, the resulting keypoints of the mesh are shown.

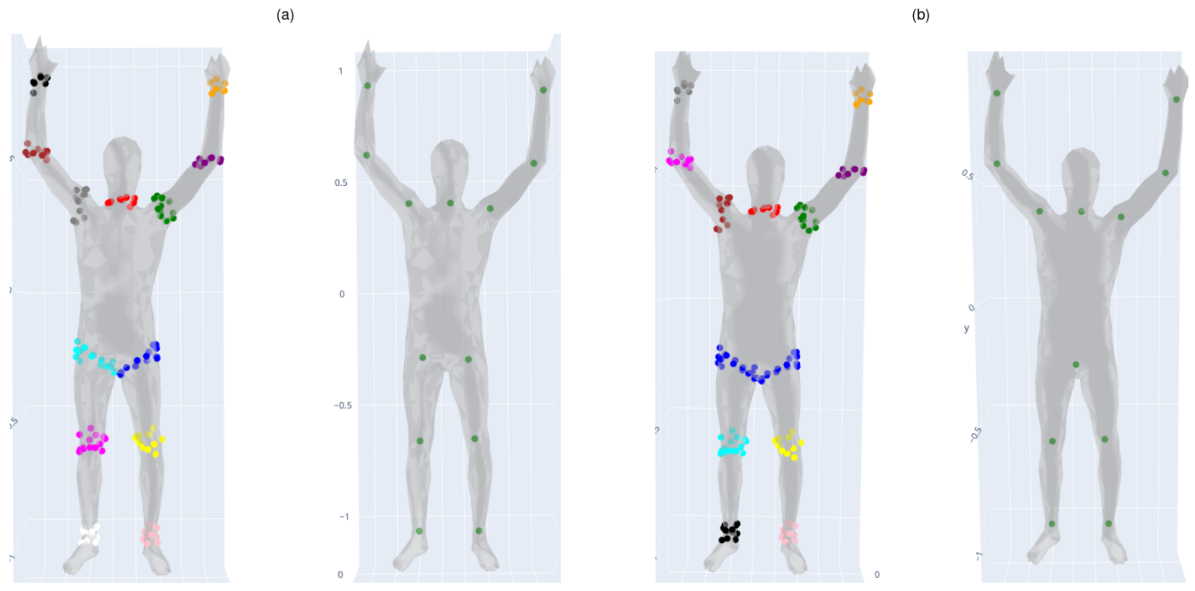

Figure 4: Illustrative example of a mesh containing 13 keypoints in subfigure *a*, and the same mesh after merging the torso-thigh connection into a single component in subfigure *b*.

4.1.3. Ambiguity in Symmetrical Keypoint Prediction

As previously discussed, the Human Segmentation Dataset provides segmentations of meshes across the eight previously mentioned classes (section 4.1). However, it lacks information regarding the lateral distinction (left/right) of the segmented parts. This omission introduces ambiguity in symmetrical keypoints, including shoulders, elbows, wrists, knees, and ankles. Consequently, in the Human Mesh Keypoints Extraction dataset, we are unable to differentiate between left and right keypoints.

4.2. Edge Features

Inspired by the architecture proposed by MeshCNN [11], our approach aims to learn the keypoint positions of a human mesh using edge features.

4.2.1. MeshCNN Edge Features

MeshCNN introduced the following set of features for each individual edge:

- Dihedral angle ϕ,
- Two inner angles α_1, α_2,
- Two edge-length ratios $\frac{|e|}{|h_1|}, \frac{|e|}{|h_2|}$.

Thus, the feature vector for each edge is represented as:

$$\overrightarrow{f_e} = \left(\phi, \alpha_1, \alpha_2, \frac{|e|}{|h_1|}, \frac{|e|}{|h_2|}\right) \in \mathbb{R}^5 \tag{1}$$

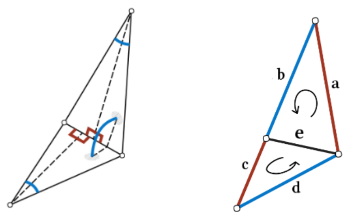

Figure 5: MeshCNN features [*left*]. Edge e and neighbors a, b, c, d [*right*]

The receptive field of an edge e is composed of both the features of e itself and the features of neighboring edges, represented as:

$$(e^1, e^2, e^3, e^4) = (|a - c|, a + c, |b - d|, b + d) \tag{2}$$

This ensures convolutional invariance with respect to the ordering of the input data.

All features of edge e and its neighboring edges e^1, e^2, e^3, e^4 are aggregated into a tensor, resulting in a dimension of 5×5, as there are 5 edges in total, and each edge has 5 features (one dihedral angle, two inner angles, and two edge-length ratios).

4.2.2. Our Edge Features

To simplify and speed up feature extraction, we opted to modify the type of edge features. Specifically, we used the following features for each individual edge:

- coordinates of the first vertex $v_1 = (x_1, y_1, z_1)$,
- coordinates of the second vertex $v_2 = (x_2, y_2, z_2)$,
- edge length $L = ||v_2 - v_1||$.

Thus, the feature vector for each edge is represented as:

$$\overrightarrow{f_e} = (x_1, y_1, z_1, x_2, y_2, z_2, L) \in \mathbb{R}^7 \tag{3}$$

In our implementation, the receptive field of an edge e is composed of both the features of e itself and the features of neighboring edges, represented as:

$$(e^1, e^2, e^3, e^4) = (|a - c|, a + c, |b - d|, b + d) \tag{4}$$

This ensures convolutional invariance with respect to the ordering of the input data.

All features of the edge e are then aggregated into a tensor, resulting in a dimension of 5×7. Thus, feature extraction for the entire mesh results in a tensor of dimensions $5 \times N \times 7$, where:

- 5: number of neighbors + 1 $\rightarrow$ channels,
- N: number of edges in the mesh $\rightarrow$ height,
- 7: number of features per edge $\rightarrow$ width.

4.3. Network and Loss

4.3.1. Network

The proposed architecture is a straightforward and efficient convolutional neural network (CNN) designed for 3D keypoints prediction from mesh edge features. Its structure is simple yet effective, consisting of three main components: convolutional layers, pooling, and fully connected layers.

The input of the network is a tensor of edge features of dimension $C \times H \times W$, where $C = 5$ is the number of neighbors of the current edge e plus 1 (the current edge), $W = 7$ is the number of features per edge, and H is the number of edges in the mesh. The output of the network is a tensor of keypoints of dimension 12×3, where 12 represents the number of keypoints of a mesh, and 3 represents the x, y, and z coordinates of each keypoint.

The architecture begins with three *convolutional layers* (`conv2d`), which progressively increase the number of filters (64, 128, 256), allowing the network to capture more complex spatial features from the input edge data. Each convolutional layer is followed by a *ReLU* activation to introduce non-linearity, with all layers maintaining spatial dimensions due to padding.

Next, the network employs an *adaptive max pooling* layer, which simplifies the feature map to a 1x1 size, condensing the global information of the mesh into a single feature vector.

Finally, the pooled features are passed through two *fully connected layers*: the first reduces the dimensionality to 128, and the second outputs the predicted keypoint coordinates. The output is reshaped to provide the 3D coordinates for each keypoint.

This simple architecture allows for efficient learning and prediction, balancing depth with computational efficiency, making it easy to train and modify for different input sizes and keypoint numbers. Figure 6 illustrates the scheme of the architecture.

4.3.2. Loss

One of the main challenges in this project is the unordered nature of the keypoints, meaning that they may appear in different sequences across various meshes. This inconsistency creates significant difficulties during the training phase, as a straightforward learning approach — such as using the Mean Squared Error (MSE) loss — proves ineffective when the keypoints lack a consistent order.

Throughout the implementation process, we experimented with three types of loss functions:

- Mean Squared Error (MSE) Loss,
- Chamfer Loss,
- Sum of Distances Loss with Hungarian Matching.

As discussed in the following sections, both the MSE loss and the Chamfer loss are inadequate for this task due to the unordered keypoints in the dataset. Consequently, the most effective approach was to use the Sum of Distances Loss with Hungarian Matching, which better handles the unordered nature of the keypoints and improves model performance.

Sum of Distances Loss with Hungarian Matching The Sum of Distances Loss with Hungarian Matching establishes an optimal one-to-one correspondence between the predicted keypoints and the ground truth keypoints. This

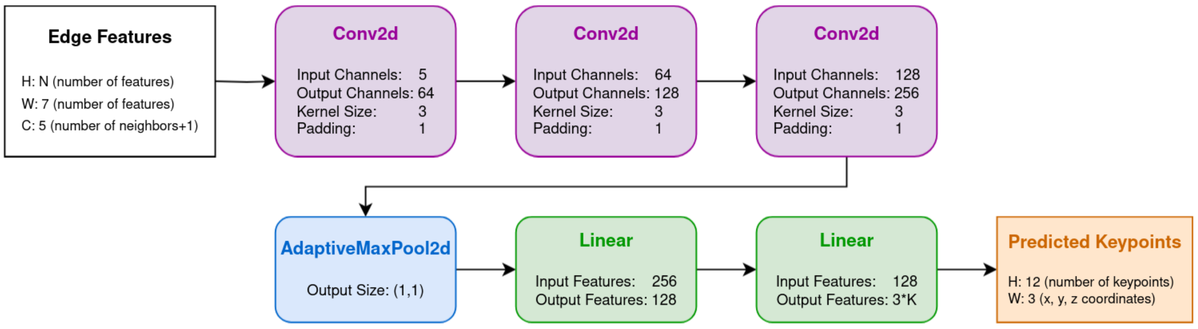

Figure 6: Proposed architecture for predicting 3D keypoints from edge features of a mesh. The input to the network is a tensor containing edge features with dimensions $C \times H \times W$, where H represents the number of edges of the mesh, $W = 7$ represents the number of edge features, and $C = 5$ represents the number of neighbors plus one (the current edge). The network consists of three convolutional layers with increasing numbers of filters, followed by an adaptive max pooling layer. The final fully connected layers predict the 3D coordinates of K keypoints, with output dimensions corresponding to $H = 12$ (number of keypoints) and $W = 3$ (representing the x, y, and z coordinates of each keypoint)

matching process is efficiently executed using the Hungarian algorithm [44], which minimizes the total pairwise distance between the predicted and actual keypoints. The final loss is calculated as the sum of these distances between the matched keypoints.

$$\mathcal{L} = \min_{\sigma \in S_n} \sum_{i=1}^{n} \|\hat{k}_{\sigma(i)} - k_i\|_2 \tag{5}$$

where:

- $\hat{k}_i \in \mathbb{R}^3$ are the predicted keypoints;
- $\hat{k}_{\sigma(i)} \in \mathbb{R}^3$ are the predicted keypoints permuted according to the optimal permutation σ;
- $k_i \in \mathbb{R}^3$ are the ground truth keypoints;
- $\sigma \in S_n$ represents a permutation of the indices $\{1, \ldots, n\}$, where n is the total number of keypoints;
- $\|\cdot\|_2$ denotes the Euclidean distance (L2 norm);
- The Hungarian algorithm is utilized to determine the optimal permutation σ that minimizes the total distance between the predicted and ground truth keypoints.

MSE Loss In the implementation workflow, our initial approach involved the use of Mean Squared Error (MSE) loss, a widely recognized metric in regression problems for quantifying discrepancies between predicted and actual values. The MSE is mathematically defined as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6}$$

where n represents the number of keypoints, y_i denotes the ground truth keypoint coordinates, and $\hat{y}_i$ represents the predicted keypoint coordinates. This loss function is particularly effective in capturing the average squared deviation, making it sensitive to larger errors due to its quadratic nature. This characteristic encourages the model to focus on minimizing significant discrepancies, which can enhance overall accuracy in many regression scenarios.

However, in the context of keypoint prediction, the unordered nature of the keypoints presents a significant challenge. Since the MSE does not accommodate this lack of order, it may yield misleading results by treating the predicted and ground truth keypoints as if they were arranged in a fixed sequence. Consequently, this limitation led to poor performance during the training phase, as the model struggled to learn meaningful representations from the unordered data.

Chamfer Loss We also experimented with Chamfer loss, a well-established metric for comparing sets of points, particularly effective in tasks such as point cloud registration. This loss quantifies the distance between two point clouds by calculating the average of the squared distances from each point in one set to its nearest neighbor in the other set, as defined by the following equation:

$$\mathcal{L}_{\text{Chamfer}} = \frac{1}{|P|} \sum_{p \in P} \min_{q \in Q} \|p - q\|^2 + \frac{1}{|Q|} \sum_{q \in Q} \min_{p \in P} \|q - p\|^2 \tag{7}$$

where P and Q are two sets of points. The primary advantage of Chamfer loss lies in its simplicity and efficiency, allowing for direct comparison of point sets without requiring explicit point matching. This feature makes it particularly suitable for unordered data, providing flexibility in point cloud alignment while avoiding rigid correspondences. However, the absence of explicit matching can also pose a limitation, as the loss may be minimized under incorrect correspondences. For instance, points from one set might be erroneously matched to points in the other set that are not their true nearest neighbors, leading to suboptimal learning outcomes.

Consequently, while Chamfer loss can effectively capture the overall structure of point clouds, it does not guarantee accurate point correspondences. This limitation can impede performance in tasks that require precise alignments, such as keypoint prediction.

5. Training and Results

The original dataset, comprising 292 meshes, has been partitioned into 80% for training, 10% for validation, and 10% for testing.

5.1. Preprocessing - Centroid Normalization and Scaling

All meshes have been preprocessed through centroid normalization and scaling to effectively eliminate any positional bias that could impact subsequent analyses or learning tasks. By centering the meshes around the origin, positional bias is removed, allowing the model to focus on the geometric shape and features rather than their spatial position. Additionally, scaling the meshes after centering preserves their original dimensions, facilitating meaningful comparisons between different meshes or configurations.

This process is easily reversible, allowing the predicted keypoints to be re-centered and scaled to match the original mesh size and position after the prediction

5.2. Training

The network was trained for 100 epochs using a batch size of 32. The Adam optimizer [45] was employed with a learning rate set to 0.001. The training process was conducted on a laptop equipped with a single Nvidia 4050 GPU, taking approximately 2 minutes and 30 seconds per epoch, including the validation step. This resulted in a total training time of about 4 hours and 15 minutes.

5.3. Results

5.3.1. Metrics

Mean Per Joint Position Error (MPJPE) The Mean Per Joint Position Error (MPJPE) is a common evaluation metric used to assess the accuracy of predicted keypoint locations in human pose estimation tasks. It is calculated by measuring the Euclidean distance between the predicted joint positions and the corresponding ground truth positions, averaging this distance across all joints. Mathematically, it is expressed as:

$$MPJPE = \frac{1}{n}\sum_{i=1}^{n}\|\hat{k}_i - k_i\|_2 \tag{8}$$

where $\hat{k}_i$ and k_i represent the predicted truth and ground truth positions of the i-th joint, respectively, and n is the total number of joints. MPJPE provides a clear indication of the overall localization accuracy, with lower values indicating better performance.

Percentage of Correct Keypoints (PCK) The Percentage of Correct Keypoints (PCK) is another widely used metric that evaluates the correctness of predicted keypoints in relation to their ground truth counterparts, based on a specified threshold distance. The PCK is calculated by determining the percentage of predicted keypoints that fall within a certain radius from their ground truth locations. Formally, it can be defined as:

$$PCK(t_{pck}) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(\|\hat{k}_i - k_i\|_2 \leq t_{pck}) \tag{9}$$

where t_{pck} is the threshold distance, $\mathbb{I}$ is an indicator function that equals 1 if the condition is met and 0 otherwise. PCK provides a more intuitive measure of accuracy, reflecting the proportion of keypoints that are considered correctly predicted within a reasonable margin, thus allowing for a more lenient evaluation in cases where exact matching is difficult.

Percentage of Correct Meshes (PCM) The Percentage of Correct Meshes (PCM) is a metric designed to assess the accuracy of a predicted mesh by leveraging the Percentage of Correct Keypoints (PCK). Specifically, a mesh is considered correct if its PCK score, calculated at a specified threshold t_{pck}, exceeds a predefined mesh threshold t_{mesh}. This provides a holistic view of mesh accuracy, ensuring that a certain percentage of keypoints must be within a small error margin for the entire mesh to be deemed correct.

Mathematically is defined as:

$$PCM(t_{mesh}, t_{pck}) = \frac{1}{B}\sum_{b=1}^{B}\mathbb{I}\left(\frac{PCK(t_{pck})_b}{N} \geq t_{mesh}\right) \tag{10}$$

where:

- B is the total number of meshes in the batch;
- N is the number of keypoints per mesh;
- $PCK(t_{pck})_b$ represent the PCK@t_{pck} for the $b-$th mesh;
- t_{mesh} is the threshold for determining if a mesh is correct;
- $\mathbb{I}(\cdot)$ is an indicator function that equals 1 if the condition is met and 0 otherwise.

5.3.2. Analytic Results

The results are presented in Table 1. Despite the limited size of the dataset and the compact architecture of the network, the performance is promising. Notably, the percentage of correct keypoints with $t_{pck} = 0.1$ is higher than 84%, indicating that more the most of the keypoints in the entire test set were accurately predicted within the specified threshold.

An equally noteworthy result is the Percentage of Correct Meshes with $t_{mesh} = 0.9$ and $t_{pck} = 0.1$, which stands at approximately 63%. This signifies that over 60% of the meshes in the test set had a highly accurate prediction of keypoints, showcasing the robustness of the model in capturing mesh-level accuracy.

These results demonstrate the model's effectiveness in keypoint prediction, even when faced with challenges such as limited data and a relatively small network.

5.3.3. Visual Results

In Figure 7, examples of keypoint predictions are shown, with ground truth keypoints in green and predicted keypoints in red. Overall, the results highlight good performance, as the majority of keypoints are accurately predicted across diverse poses. In many instances, the predicted keypoints closely match the ground truth, indicating that the model successfully captures the geometric structure of the human mesh based on the edge features tensor.

However, some imperfections are observed, particularly in more challenging poses. For instance, in Figure 7e, the prediction for the right knee is noticeably less accurate, which highlights the difficulty the model encounters when predicting keypoints on complex or non-standard postures. This suggests that while the model can generalize well in most cases, it may struggle with outlier poses or cases where there are occlusions or unusual limb orientations.

Metric	Result
Sum of Distances Loss with Hungarian Matching	0.6311
Mean Per Joint Position Error (MPJPE)	0.0526
Percentage of Correct Keypoints with $t_{pck} = 0.1$ (PCK@0.1)	0.8417
Percentage of Correct Keypoints with $t_{pck} = 0.05$ (PCK@0.05)	0.6778
Percentage of Correct Meshes with $t_{mesh} = 1.0, t_{pck} = 0.1$ (PCK@0.1 = 1.0)	0.4667
Percentage of Correct Meshes with $t_{mesh} = 1.0, t_{pck}0.05$ (PCK@0.05 = 1.0)	0.2333
Percentage of Correct Meshes with $t_{mesh} = 0.9, t_{pck} = 0.1$ (PCK@0.1 $\geq$ 0.9)	0.6333
Percentage of Correct Meshes with $t_{mesh} = 0.9, t_{pck}0.05$ (PCK@0.05 $\geq$ 1.0)	0.4333

Table 1
Results of the proposed model. The Sum of Distances Loss and the Mean Per Joint Position Error (MPJPE) should be interpreted in the context of the preprocessing applied, which is centroid normalization and scaling of the mesh.

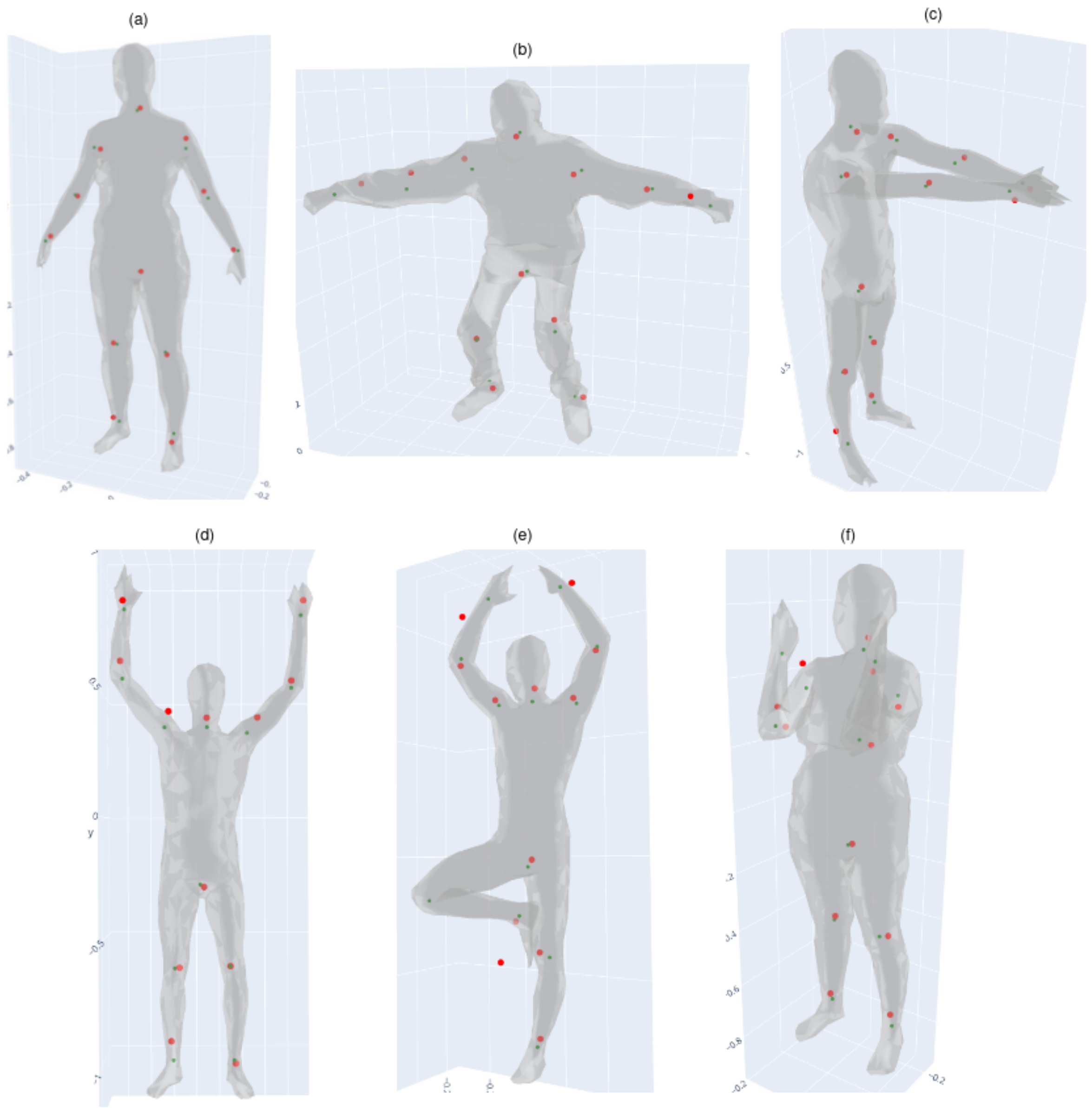

Figure 7: Visual results are presented with ground truth keypoints shown in green, and predicted keypoints highlighted in red.

6. Conclusion

In this project, we presented a novel approach for 3D human mesh keypoint extraction, with one of the key contributions being the development of a new dataset — Human Mesh Keypoints Extraction — derived from the Human Body Segmentation dataset. This dataset addresses the scarcity of suitable resources for 3D human pose estimation, particularly those based on mesh structures.

We proposed a straightforward yet highly effective neural network architecture that uses mesh edge features to predict 12 keypoints corresponding to critical human joints. The simplicity of the architecture enables efficient training and accurate predictions while keeping computational demands low. A central point of our method is the use of Hungarian Matching in the loss function to resolve the keypoint association problem caused by the unordered nature of keypoints. This technique ensures the correct matching of keypoints between predictions and ground truth, significantly enhancing model performance.

The results validate the potentials of our approach, showing high accuracy across key evaluation metrics, including Mean Per Joint Position Error (MPJPE) and Percentage of Correct Keypoints (PCK). Despite challenges such as the limited size of the dataset and the complexity of certain poses, the model delivered promising results, highlighting its potential for wider applications in 3D human pose estimation. Future work could focus on expanding the dataset and refining the architecture to improve performance on more complex and varied human poses.

References

[1] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, A. Baskurt, Sequential deep learning for human action recognition, in: International Workshop on Human Behavior Unterstanding, 2011. URL: https://api.semanticscholar.org/CorpusID:12591063.

[2] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, D. Tran, Detect-and-track: Efficient pose estimation in videos (2017). doi:10.48550/arXiv.1712.09184.

[3] Z. Liu, S. Wu, S. Jin, Q. Liu, S. Lu, R. Zimmermann, L. Cheng, Towards natural and accurate future motion prediction of humans and animals, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 9996–10004. URL: https://api.semanticscholar.org/CorpusID:157061625.

[4] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, Openpose: Realtime multi-person 2d pose estimation using part affinity fields, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (2021) 172–186. doi:10.1109/TPAMI.2019.2929257.

[5] Çiçek, A. Abdulkadir, S. Lienkamp, T. Brox, O. Ronneberger, 3d u-net: Learning dense volumetric segmentation from sparse annotation (2016). doi:10.48550/arXiv.1606.06650.

[6] D. Maturana, S. Scherer, Voxnet: A 3d convolutional neural network for real-time object recognition, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 922–928. doi:10.1109/IROS.2015.7353481.

[7] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis, in: Proceedings of the European Conference on Computer Vision (ECCV), 2020. URL: http://arxiv.org/abs/2003.08934v2.

[8] J. J. Park, P. R. Florence, J. Straub, R. A. Newcombe, S. Lovegrove, Deepsdf: Learning continuous signed distance functions for shape representation, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 165–174. URL: https://api.semanticscholar.org/CorpusID:58007025.

[9] C. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: Neural Information Processing Systems, 2017. URL: https://api.semanticscholar.org/CorpusID:1745976.

[10] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: Going beyond euclidean data, IEEE Signal Processing Magazine 34 (2017) 18–42. doi:10.1109/MSP.2017.2693418.

[11] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, D. Cohen-Or, Meshcnn: a network with an edge, ACM Trans. Graph. 38 (2019). URL: https://doi.org/10.1145/3306346.3322959. doi:10.1145/3306346.3322959.

[12] C.-H. Chen, D. Ramanan, 3d human pose estimation = 2d pose estimation + matching (2016). doi:10.48550/arXiv.1612.06524.

[13] N. Garau, N. Bisagno, P. Br'odka, N. Conci, Deca: Deep viewpoint-equivariant human pose estimation using capsule autoencoders, 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (2021) 11657–11666. URL: https://api.semanticscholar.org/CorpusID:237213569.

[14] J. Martinez, M. R. I. Hossain, J. Romero, J. Little, A simple yet effective baseline for 3d human pose estimation, 2017, pp. 2659–2668. doi:10.1109/ICCV.2017.288.

[15] J. Tierny, J.-P. Vandeborre, M. Daoudi, 3D Mesh Skeleton Extraction Using Topological and Geometrical Analyses, in: 14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2006), Tapei, Taiwan, 2006, p. s1poster. URL: https://hal.science/hal-00725576.

[16] F. Gong, C. Kang, 3d mesh skeleton extraction based on feature points, in: 2009 International Conference on Computer Engineering and Technology, volume 1, 2009, pp. 326–329. doi:10.1109/ICCET.2009.71.

[17] O. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, T.-Y. Lee, Skeleton extraction by mesh contraction, ACM Trans. Graph. 27 (2008). doi:10.1145/1360612.1360643.

[18] H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, Y. Lipman, Convolutional neural networks on surfaces via seamless toric covers, ACM Trans. Graph. 36 (2017). URL: https://doi.org/10.1145/3072959.3073616. doi:10.1145/3072959.3073616.

[19] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, J. Davis, Scape: shape completion and animation of people, in: ACM SIGGRAPH 2005 Papers, SIGGRAPH '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 408–416. URL: https://doi.org/10.1145/1186822.1073207. doi:10.1145/1186822.1073207.

[20] F. Bogo, J. Romero, M. Loper, M. J. Black, Faust: Dataset and evaluation for 3d mesh registration, in: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '14, IEEE Computer Society, USA, 2014, p. 3794–3801. URL: https://doi.org/10.1109/CVPR.2014.491. doi:10.1109/CVPR.2014.491.

[21] D. Vlasic, I. Baran, W. Matusik, J. Popović, Articulated mesh animation from multi-view silhouettes, ACM Trans. Graph. 27 (2008) 1–9. URL: https://doi.org/10.1145/1360612.1360696. doi:10.1145/1360612.1360696.

[22] Adobe, Adobe fuse 3d characters, https://www.mixamo.com/, 2016.

[23] D. Giorgi, S. Biasotti, L. Paraboschi, Shape retrieval contest 2007: Watertight models track, SHREC competition 8 (2008).

[24] E. Kalogerakis, A. Hertzmann, K. Singh, Learning

3d mesh segmentation and labeling, ACM Trans. Graph. 29 (2010). URL: https://doi.org/10.1145/1778765.1778839. doi:10.1145/1778765.1778839.

[25] A. Toshev, C. Szegedy, Deeppose: Human pose estimation via deep neural networks, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1653–1660. doi:10.1109/CVPR.2014.214.

[26] X. Sun, S. Jiaxiang, S. Liang, Y. Wei, Compositional human pose regression, Computer Vision and Image Understanding 176-177 (2017). doi:10.1016/j.cviu.2018.10.006.

[27] L. Qiu, X. Zhang, Y. Li, G. Li, W. Xiaojun, Z. Xiong, X. Han, S. Cui, Peeking into Occluded Joints: A Novel Framework for Crowd Pose Estimation, 2020, pp. 488–504. doi:10.1007/978-3-030-58529-7_29.

[28] S.-E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional pose machines, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 4724–4732. URL: https://api.semanticscholar.org/CorpusID:163946.

[29] Y. Cai, Z. Wang, Z. Luo, B. Yin, A. Du, H. Wang, X. Zhou, E. Zhou, X. Zhang, J. Sun, Learning delicate local representations for multi-person pose estimation, 2020. doi:10.48550/arXiv.2003.04030.

[30] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, J. Sun, Cascaded pyramid network for multi-person pose estimation, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7103–7112. doi:10.1109/CVPR.2018.00742.

[31] K. Sun, B. Xiao, D. Liu, J. Wang, Deep high-resolution representation learning for human pose estimation, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 5686–5696. doi:10.1109/CVPR.2019.00584.

[32] Z. Geng, S. ke, B. Xiao, Z. Zhang, J. Wang, Bottom-up human pose estimation via disentangled keypoint regression, 2021. doi:10.48550/arXiv.2104.02300.

[33] A. Newell, J. Deng, Z. Huang, Associative embedding:end-to-end learning for joint detection and grouping (2016). doi:10.48550/arXiv.1611.05424.

[34] X. Sun, B. Xiao, F. Wei, S. Liang, Y. Wei, Integral human pose regression, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 529–545.

[35] S. Park, J. Hwang, N. Kwak, 3d human pose estimation using convolutional neural networks with 2d pose information, in: Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14, Springer, 2016, pp. 156–169.

[36] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, M. J. Black, Smpl: A skinned multi-person linear model, in: Seminal Graphics Papers: Pushing the Boundaries, Volume 2, 2023, pp. 851–866.

[37] M. Kocabas, S. Karagoz, E. Akbas, Self-supervised learning of 3d human pose using multi-view geometry, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 1077–1086.

[38] H. Qiu, C. Wang, J. Wang, N. Wang, W. Zeng, Cross view fusion for 3d human pose estimation, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 4342–4351.

[39] R. Ogniewicz, M. Ilg, Voronoi skeletons: theory and applications, in: Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1992, pp. 63–69. doi:10.1109/CVPR.1992.223226.

[40] G. Borgefors, I. Nyström, G. S. di Baja, Computing skeletons in three dimensions, Pattern Recognit. 32 (1999) 1225–1236. URL: https://api.semanticscholar.org/CorpusID:17056781.

[41] Y. Shinagawa, T. Kunii, Constructing a reeb graph automatically from cross sections, IEEE Computer Graphics and Applications 11 (1991) 44–51. doi:10.1109/38.103393.

[42] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, G. Gkioxari, Accelerating 3d deep learning with pytorch3d, arXiv:2007.08501 (2020).

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, 2019. URL: https://arxiv.org/abs/1912.01703. arXiv:1912.01703.

[44] H. W. Kuhn, The Hungarian Method for the Assignment Problem, Naval Research Logistics Quarterly 2 (1955) 83–97. doi:10.1002/nav.3800020109.

[45] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).