Neural Approaches for 3D Pose Estimation from 3D Data

Gali HOD *[1], Tal BARAMI [1,2], Michael KOLOMENKIN *[1]
[1] Playtika LTD., Herzliya, Israel;
[2] Department of Computer Science, Ben-Gurion University of the Negev, Beersheba, Israel

https://doi.org/10.15221/23.42

Abstract

Understanding human pose is a fundamental component of many forms of art, including sculpture, painting, drawing, and animation. Software that can accurately capture and represent the human pose is essential for creating realistic and expressive works of both traditional and digital art. However, to the best of our knowledge, there is currently no open-source code available for deep learning-based pose estimation from static 3D data. While there exist many "classical" pre-deep learning methods for this task, they have a significant drawback: they are not differentiable, making them difficult to incorporate into subsequent deep learning pipelines. We put special emphasis on integration with deep learning pipelines since they are the cornerstone of modern creative systems. In this work, we propose and implement two methods for human pose estimation based on neural networks. The first method leverages part segmentation to classify the body part of each point and estimates the body joints based on neighboring parts. The second method estimates joints directly from point clouds. Our code will be made available on GitHub.

Keywords: Human 3D pose estimation, neural network, point cloud, mesh, skeleton reconstruction

1. Introduction

Human pose estimation deals with estimation of the spatial locations of key joints on the human body from an image, video, or a 3D scan. Human pose estimation is a crucial component of any creative software because it enables it to understand and represent human movement and expression. By accurately estimating the position and orientation of the human body and its limbs, software can create lifelike animations, generate virtual characters that mimic human behavior, and interact with users in more natural and intuitive ways. Moreover, understanding human pose allows creative software to be used in a wide range of applications, from video games and animation to virtual reality and augmented reality experiences. Whether it is creating lifelike characters for a movie or designing interactive applications for virtual and augmented reality, accurate human pose estimation is essential for achieving realistic and engaging results [14].

The existing human pose estimation techniques can be categorized into several groups based on two factors. The first factor is the input type, which can be 2D, 2.5D, or 3D, depending on whether the input is images or videos and depth images, or point clouds and meshes, respectively. The second factor is the solution type, which is whether it is based on machine learning or not.

Estimating a 3D pose from 2D data is a more complicated task compared to extracting a 3D pose from 3D data, as the latter naturally holds more information. However, most of the existing research is dedicated to extracting poses from regular or depth images rather than from meshes or point clouds [15, 8 ,13, 6, 16]. This is so since the 2D media is the most widespread source of data. With recent advances in mobile technology turning any phone into a high-quality 3D scanner, the amount of 3D data is constantly increasing. The need for tools to process the data is increasing too. In this work, we focus on human pose estimation from 3D data. There exist methods for pose estimation from point clouds, for instance see [15] for an excellent review of what was published before 2022.

The second factor is the solution type - is it based on machine learning or not. The reason why we look at the solution type is that currently pose estimation methods often need to be incorporated into deep learning pipelines [12]. Thus, they have to allow back propagation. "Classical" pose estimation method are typically not derivable and their incorporation into deep learning pipelines requires a lot of effort [9, 3, 11]. The ability to use a model in a deep learning pipeline is critical for any modern system. We are aware of two machine learning based methods for pose estimation from point clouds. The first one is described in [5], which reconstructs a pose from a point cloud. However, its code has not been made publicly available yet. The second method is described in [1] and it does have an open-source implementation. However, this method is a multi-view, event-based approach that does not directly solve the straightforward problem of pose reconstruction.

* {galih,michaelko}@playtika.com

At this work we implement and examine two different approaches for pose estimation from 3D point clouds and meshes. The first approach reconstructs the skeleton from meshes in two steps. Initially, the input mesh is divided into individual body parts through segmentation. Next, joints, which connect different body parts, are identified. A separate, iterative process is used to calculate the spine joints. This method can be adjusted to estimate the skeletons of not only humans but also animals, with very small effort.
The second method reconstructs the skeleton from 3D point clouds directly using a neural network. The network estimates the relative joint positions and rotations similar to AMASS [7]. The network uses pointNet++ [10] for 3D feature extraction.
We evaluate both methods by calculating the root mean squared error (RMSE) between the estimated and ground-truth pose and by visualizing the ground truth and the estimated point clouds. The project's code will be made available at our GitHub page providing pose estimation models of the two different methods including an evaluation and comparison between them. The code is straightforward to use, does not have special dependencies and can be easily applied to a variety of tasks, such as pose-controllable avatar generation and animation.

2. Our Approach

In this section we present two methods for 3D pose estimation from 3D data. The methods are called “segmentation-based” and “direct”. Both methods are based on neural networks and, thus, can be used in deep learning pipelines.
The segmentation-based method finds the skeleton in two steps. First it segments the body parts and then it computes the location of joints from the body parts. The direct method trains a neural network to find the joints directly.
We implemented the methods so that to support as input both meshes and point clouds. The segmentation-based method uses meshes as input, while the direct method uses point clouds. In general, it is possible to convert point clouds into meshes and vice versa and we may add this option in the future. Meanwhile, the reader may use MeshLab [2] for conversion. Note that the neural network working on point clouds assumes uniform distribution of points.
Below we describe the methods in more detail.

2.1. Segmentation-Based Method

The method solves skeleton estimation problem in two steps. First, it segments the input mesh by allocating every point to a body part. Then it applies anatomical knowledge to the detected body parts to find the location of joints.
The segmentation step uses a neural network model presented in [4]. We made several small changes to the model dimensions and data types to fit our data.
The model is trained on our dataset, see Implementations for details.
The following body parts are detected by our segmentation model: the head, torso, arms (left & right), forearms (left & right), hands (left & right), thighs (left & right), calves (left & right) and feet (left & right). The body parts are demonstrated in Figure 1.

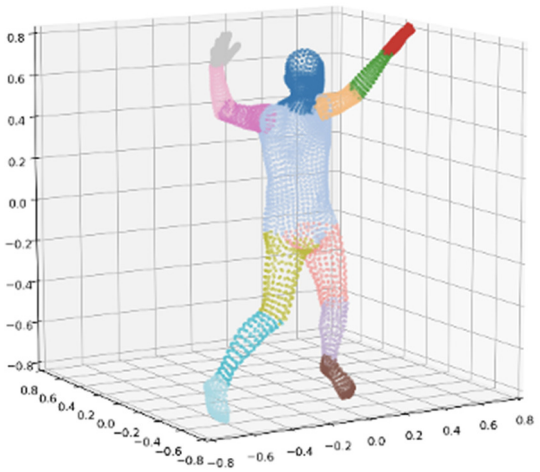

Fig. 1. Body parts visualization. All the body parts used for segmentation are visualized on a point cloud - the head, torso, arms (left & right), forearms (left & right), hands (left & right), thighs (left & right), calves (left & right) and feet (left & right).

The locations of joints can be found from the segmented mesh by applying some common-sense anatomical knowledge. We show the accuracy of the resulted locations in Results. Below we describe how the locations are computed. The computation details depend on the type of the joint:

1. Connecting joints: These are the joints that connect two different body parts. i.e., neck, shoulders, elbows, wrists, hips, knees, and ankles. We look for points that belong to a body part *a* and has close neighbors in the neighboring body part *b*. The set of these points is a ring of points that surrounds the joint. The mean value of this ring is the joint location. These are the majority of joints - 2, 5, 8, 1, 4, 7, 14, 17, 19, 21, 13, 16, 18, 20, and 12 in Figure 3.
2. End joints: These joints are connected to a single limb. I.e., head, hands, and feet. In this case, it is sufficient to find the mean value of the points that belong to the limb to receive a good approximation. These are the joints 10, 11, 22, 23, and 15 in Figure 3.
3. Spine joints: Joints that connect the upper-body limbs with the lower-body limbs. These are the joints 0, 3, 6, 9 in Figure 3. The pelvis (0) is reconstructed using the mean value of the reconstructed hips. The neck (9) is reconstructed as the average of shoulders. To reconstruct the spine joints 3 and 6, we compute an approximation to the median of the torso. We do it in two steps. First, we calculate the vector that connects the neck to the pelvis. Then we compute the average of the torso points perpendicular to the vector in the third and two third of the distance of the vector, see Figure 2.

The joints are demonstrated in Figure 3.

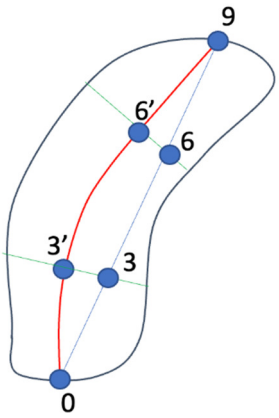

Fig. 2. Visualization of spine computation. The red line is the real spine. The blue vector connects the pelvis 0 with the neck 9. The points 3 and 6 are the initial estimations of the spine and the points 3' and 6' are the final estimations.

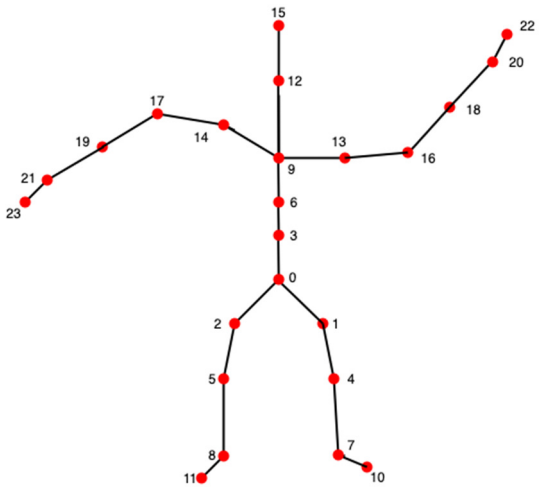

Fig. 3. Skeleton with joints illustration.

2.2. Direct Method

The direct method computes the joints directly by using a neural network. The neural network is trained on AMASS. It is based on PointNet++ as the basic feature detection.
The Direct method extracts quaternion representation for each joint, considering head movements in all directions, including yaw, pitch, and roll. However, the Segmentation method extracts the joints' locations as described in Segmentation Based Method, and it does not consider the yaw movement of the head. Consequently, the skeleton remains unchanged as the head moves about the yaw axis.
It is worth mentioning that the Segmentation based method can be easily adjusted for other skeletons by adjusting the second step of the method. In contrast, this is not possible with the Direct method since it relies on AMASS to move from the regressed pose code to point cloud and skeleton. Therefore, adapting the Direct method to other skeletons is not easy because AMASS is only familiar with human samples.

3. Results

Figures 4 and 5 show graphs summarizing the MSE value per joint for each of the methods, along with the mean error value. The meshes and point clouds used are normalized to a unit sphere, so the average RMSE value, relative to the mesh/point cloud size, is 1.52% and 2.16% for each method respectively.
For the first method, the largest average error is observed for the 'spine1' joint, which is around three times the average of all errors. However, in percentage terms, it is still relatively small.
For the second method, no single joint stands out with relatively high error compared to others. However, there is a pattern in which the joints closest to the pelvis obtain the smallest error.
The average error of the first method is smaller compared to the second method. Yet, in the second method, the errors of the pelvis and nearby joints are smaller, which may contribute to a more visually pleasing result.

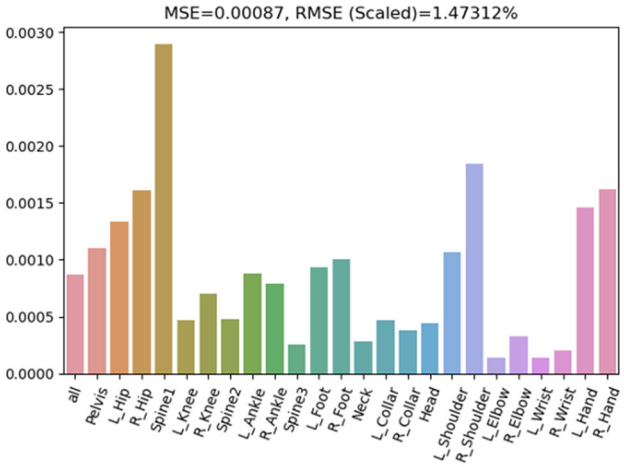

Fig. 4. Segmentation method: Mean MSE values per joint and mean error value of all joints.

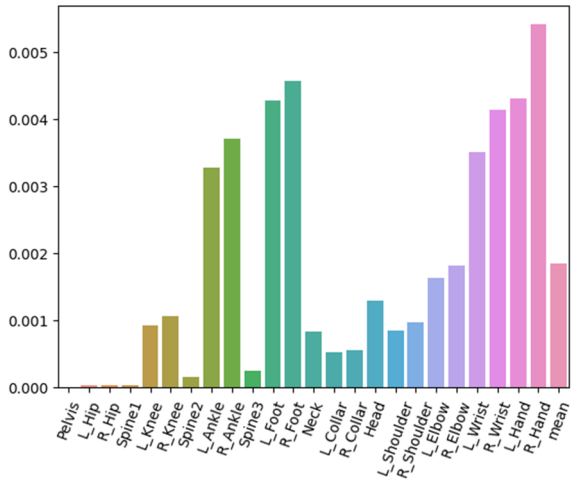

Fig. 5. Direct method: Mean MSE values per joint and mean error value of all joints.

3.1. Segmentation Based Method Results

In Figure 6 examples of estimated and ground-truth (GT) skeletons are presented, placed inside a given sample with GT and estimated segmentation, respectively. It is evident that the utilization of segmentation in the skeleton reconstruction tasks produced visually pleasing results for the most part. The most noticeable difference is observed at the 'spine1' joint, which is due to the spine not being straight and the use of only an approximation to find it.

3.2. Direct Method Results

Figure 7 show estimated and GT point clouds. The point clouds are projected and displayed in 3 different viewpoints for better observation. The qualitative results demonstrate an eye pleasing result, a main concern in creative tasks. A small difference in the hand joint is observed.
We choose to display the results of this method in such manner as it demonstrates the use of the estimated pose in generating a 3D human with a specific, given pose. A possible use case of the proposed method would be using it to enforce human pose controllability in 3D generation tasks.

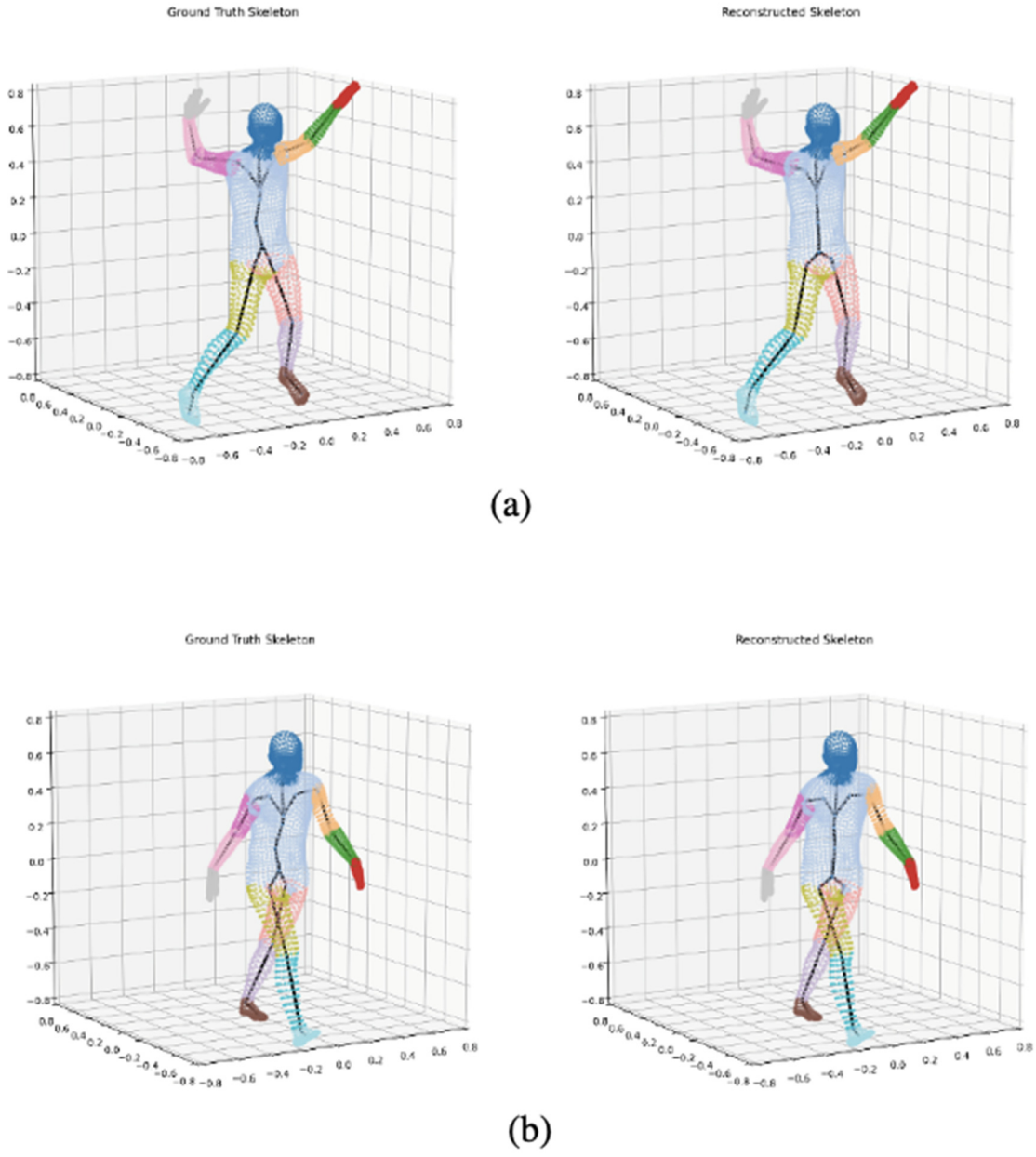

Fig. 6. Segmentation method: Estimated (right) and ground truth (left) skeletons and the corresponding segments.

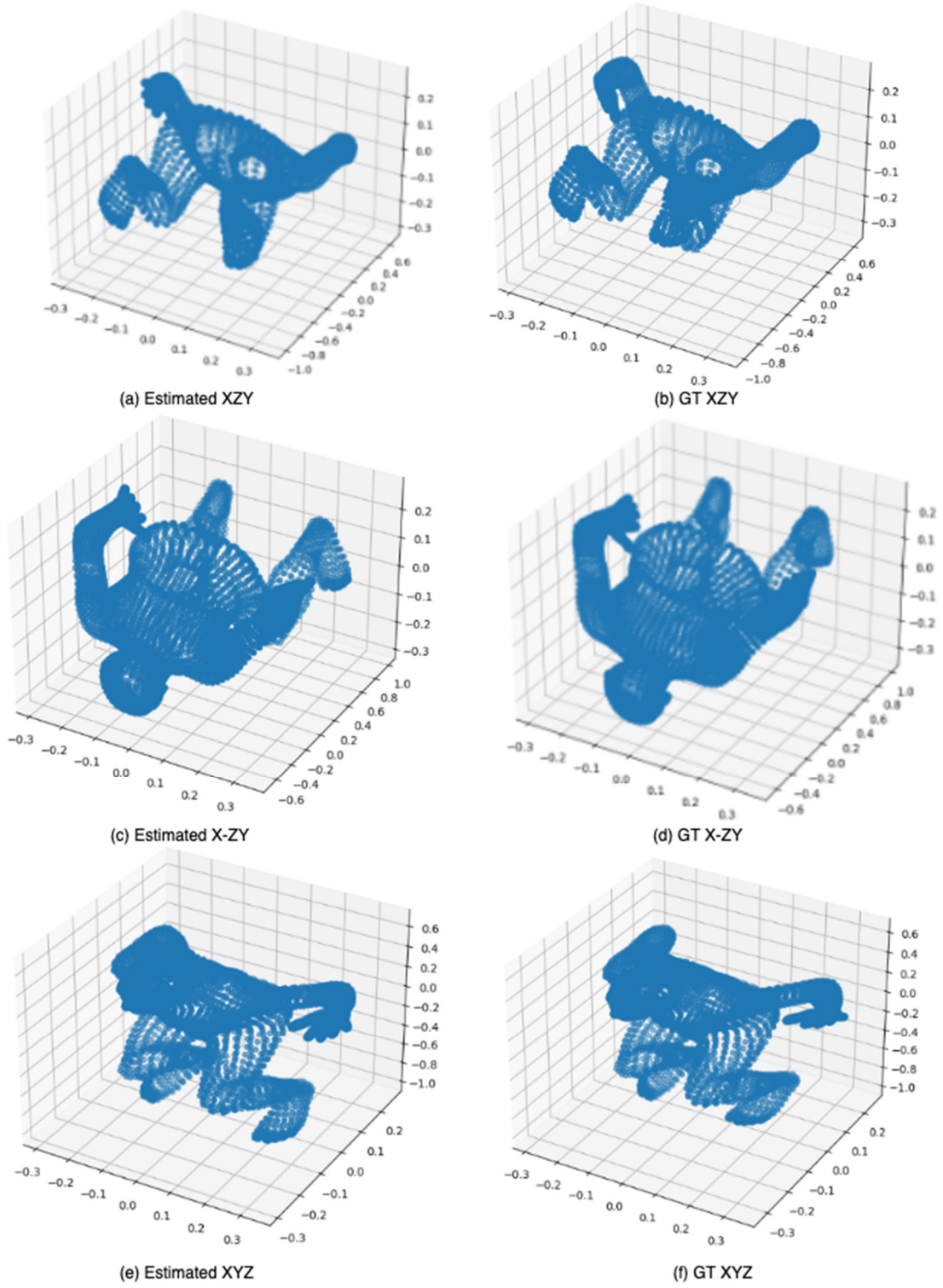

Fig. 7. Direct method: Estimated (left) and ground truth (right) point cloud projected to different viewpoints.

4. Conclusions

Human pose estimation is a basic part of any software dealing with human representation. This paper presents two fast and accurate methods for human pose estimation from 3D data, which can be easily integrated into deep learning pipelines. The authors plan to provide reliable and easy to use open-source implementation of the methods.

In future work, the authors suggest extending the methods to non-human skeletons and integrating them with popular game building tools such as Unity.

References

[1] J. Chen, H. Shi, Y. Ye, K. Yang, L. Sun and K. Wang, "Efficient Human Pose Estimation via 3D Event Point Cloud," 2022 International Conference on 3D Vision (3DV), Prague, Czech Republic, 2022, pp. 1-10, doi: https://doi.org/10.1109/3DV57658.2022.00023.

[2] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia MeshLab: an Open-Source Mesh Processing Tool Sixth Eurographics Italian Chapter Conference, page 129-136, 2008

[3] M. Dantone, J. Gall, C. Leistner and L. Van Gool, "Human Pose Estimation Using Body Parts Dependent Joint Regressors," 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 2013, pp. 3041-3048, doi: https://doi.org/10.1109/CVPR.2013.391.

[4] A. Fries Deep Learning on 3D Meshes, Stanford CS224W GraphML Tutorials, https://medium.com/stanford-cs224w/deep-learning-on-3d-meshes-9608a5b33c98, 2015

[5] H. Jiang, J. Cai and J. Zheng, "Skeleton-Aware 3D Human Shape Reconstruction From Point Clouds," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 5430-5440, doi: https://doi.org/10.1109/ICCV.2019.00553.

[6] M. Kocabas, N. Athanasiou and M. Black, "VIBE: Video Inference for Human Body Pose and Shape Estimation," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020 pp. 5252-5262. doi: https://doi.org/10.1109/CVPR42600.2020.00530

[7] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll and M. Black, "AMASS: Archive of Motion Capture As Surface Shapes," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 5441-5450, doi: https://doi.org/10.1109/ICCV.2019.00554.

[8] Newell, A., Yang, K., Deng, J. (2016). Stacked Hourglass Networks for Human Pose Estimation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9912. Springer, Cham. https://doi.org/10.1007/978-3-319-46484-8_29

[9] L. Pishchulin, M. Andriluka, P. Gehler and B. Schiele, "Poselet Conditioned Pictorial Structures," 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 2013, pp. 588-595, doi: https://doi.org/10.1109/CVPR.2013.82

[10] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, 'Pointnet++: Deep hierarchical feature learning on point sets in a metric space *Advances in neural information processing systems* 30 (2017)., doi: https://doi.org/10.48550/arXiv.1706.02413

[11] B. Sapp, C. Jordan and B. Taskar, "Adaptive pose priors for pictorial structures," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 2010, pp. 422-429, doi: https://doi.org/10.1109/CVPR.2010.5540182.

[12] A. Shoshan, N. Bhonker, I. Kviatkovsky and G. Medioni, "GAN-Control: Explicitly Controllable GANs," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 14063-14073, doi: https://doi.org/10.1109/ICCV48922.2021.01382.

[13] D. Tome, C. Russell and L. Agapito, "Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 5689-5698, doi: https://doi.org/10.1109/CVPR.2017.603.

[14] Wang, J., Tan, S., Zhen, X., Xu, S., Zheng, F., He, Z., & Shao, L. (2021). Deep 3D human pose estimation: A review. Computer Vision and Image Understanding, 210, [103225]. https://doi.org/10.1016/j.cviu.2021.103225

[15] T. Xu, D. An Y. Jia, Y. Yue A Review: Point Cloud-Based 3D Human Joints Estimation. Sensors (Basel). 2021;21(5):1684. doi: https://doi.org/10.3390/s21051684

[16] Ye, H., Zhu, W., Wang, C., Wu, R., and Wang, Y., "Faster VoxelPose: Real-time 3D Human Pose Estimation by Orthographic Projection", arXiv e-prints, 2022. doi: https://doi.org/10.48550/arXiv.2207.10955.