Supplementary Material: Skeleton2Mesh: Kinematics Prior Injected Unsupervised Human Mesh Recovery

Zhenbo Yu[1,2]*, Junjie Wang[1,2]*, Jingwei Xu[1,2], Bingbing Ni[1,2]†
Chenglong Zhao[1,2], Minsi Wang[1,2], Wenjun Zhang[1,2]
[1]Shanghai Jiao Tong University, [2]Shanghai Key Lab of Digital Media Processing & Transmission
{yuzhenbo, dreamboy.gns, xjwxjw, nibingbing, cl-zhao,mswang1994, zhangwenjun}@sjtu.edu.cn

1. Detailed Architecture

The baseline of unsupervised 3D pose estimation in our work is based on [2]. The basic building blocks of all our modules are residual blocks, which can be divided into two types: Res1 Block, and Res2 Block (see Fig. 2).

Residual Block The building block of our network architecture (the residual block illustrated in Fig. 1) is generally inspired from [5]. Res2 Block has two branches: one branch consists of two convolutional layers followed by batch normalization, ReLU activation and dropout layers sequentially; the other branch is a shortcut processing the concatenation of upstream features and input 2D joints. The kernel size and stride of each layer is equal to 1. Res1 Block is similar to Res2 Block, but with some extra layers in the main branch to process input 2D joints. All features are of dimension 1024 in residual block except the skip layer.

Network architecture As is shown in Fig. 2, in terms of our pose estimation baseline, we use five residual blocks for the lifting network and two blocks for the discriminator. Specifically, we remove the batch norm layers from the residual block in discriminator to maintain the network stable in the training stage. Besides, we use resnet18[4] as the feature extractor from silhouettes. Four parallel fully connected layers are used to learn body shapes, perform pose refinement, learn camera intrinsics and learn viewpoints. We denote residual blocks above as "n $\times$ Res Block".

Geometric Random Rotation. Geometric random rotation in our paper is similar to [2]. Before being passed to the pipeline for the second time, the 3D poses need to be randomly rotated, shifted and then projected onto 2D plane. During this process, we first uniformly sample the angles from predefined ranges. In this work, we set the ranges to $[-7\pi/9, 7\pi/9]$, $[-\pi/9, \pi/9]$, $[-\pi/18, \pi/18]$ for y, x, z axes respectively. We then use rodrigues formula to get corresponding rotation matrices, multiply them in order(e.g. y, x, z) and obtain final rotation matrix $\mathbf{R}$.

*equal contribution
†corresponding author

Baselines	Adv. Loss	Azimuth	MPJPE	PMPJPE
Baseline[2]	BCE	π	-	68.0
*Baseline[2]	BCE	π	119.3	64.3
Variant	LS	π	116.7	58.6
Final Baseline	LS	$7/9 * \pi$	**112.6**	**54.6**

Table 1: The analysis on baseline performance. * indicates our implementation according to [2]. Based on our implementation, we further make some improvements, as listed in the table. **Adv Loss:** adversarial loss function. **BCE:** binary cross entropy loss. **LS:** least square loss. **Azimuth:** the maximum azimuth in geometric random rotation.

Analysis on baseline performance. Without access to the source code of [2], we re-implement the baseline used in [2]. As shown in Tab. 1, the same baseline based on our implementation is better than that in [2]. Furthermore, we employ a different adversarial loss function (Variant), which further facilitates the performance of the network. Finally, we carefully set the ranges in geometric random rotations. We empirically find that using a smaller maximum azimuth($7\pi/9$ instead of π in [2]) results in better performance. Rotations around x and z axes are limited to $\pi/9$ and $\pi/18$ respectively. More details in our implementation can be seen in our released code.

2. Differentiable Inverse Kinematics

In this section, we describe the detailed matching process of all the ten local 3D rotations including right elbow, which is introduced in the main body of our paper. Note that the ten units (e.g., right hip, right knee, left hip, left knee, right shoulder, right elbow, left shoulder, left elbow, neck, and spine) are almost at the same position in different datasets (see Fig. 3), thus the DIK module is able to generalize to all the datasets with different topologies.

As is illustrated in Fig. 4, 3D joints (blue circle) in 3D skeleton are utilized to match the corresponding local 3D rotations (orange circle) in $\boldsymbol{\theta}_{main}$ via DIK module. 3D joints (red circle) in 3D skeleton (including head, hand, and

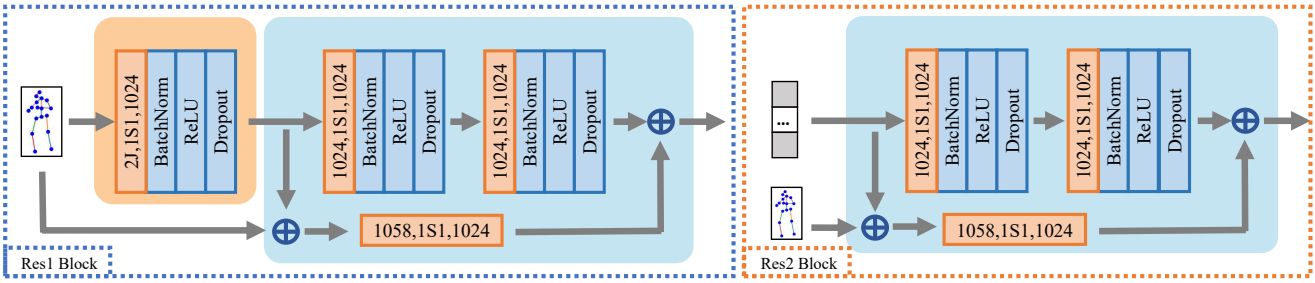

Figure 1: The detailed architecture of the Residual Block including Res1 Block and Res2 Block. The input consists of 2D keypoints with J=17 joints. Convolutional layers are in red where $2J$, $1S1$, 1024 denote $2J$ input channels, kernels of size 1 with stride 1, and 1024 output channels. And the skip layer $1058, 1S1, 1024$ denotes 1058 input channels, kernels of size 1 with stride 1, and 1024 output channels.

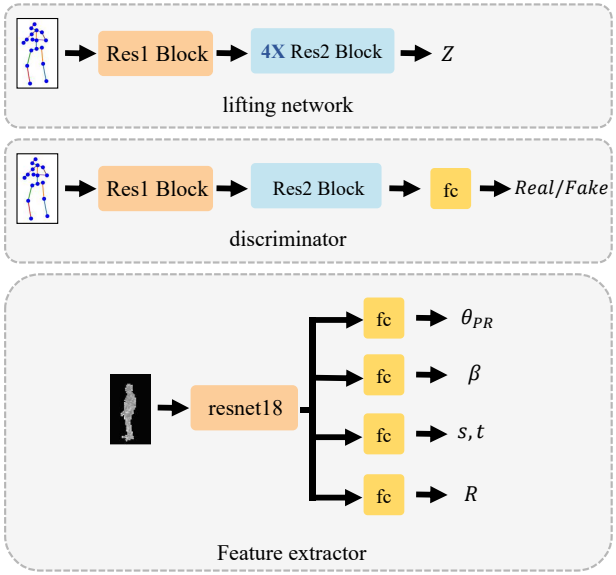

Figure 2: The detailed architecture of the lifting network, the discriminator and the shape matching branch. fc denotes fully connected layer with 1024 dimension features. $\boldsymbol{\theta}_{PR}$ indicates five local 3D rotations (i.e., one in the head, two in hands, and two in feet). $\boldsymbol{\beta}$ represents the shape parameters of SMPL model. s, t and $\mathbf{R}$ denote the camera intrinsics and the global 3D rotation, respectively.

foot) lack sufficient kinematics constraint (please refer to DIK module), thus we use PR module to learn the suitable local 3D rotations from silhouettes. Therefore, the root joint (yellow circle) is also trained to obtain the corresponding global 3D rotation in $\boldsymbol{\theta}_{main}$. Finally, some local 3D rotations (red circle) in $\boldsymbol{\theta}_{other}$ have little effect on SMPL. Concretely, four local 3D rotations in the body are almost unchanged, and the four ones in the endpoint, which represents fingers and toes, are generally ignored in human mesh recovery. To this end, we do not do any matching operations these local 3D rotations (red circle) and set these ones to the default value in SMPL.

We define the ten unit mapping equations by differentiable inverse kinematics manually (see Fig. 5), and the definition of coordinate system is exactly similar as SMPL based on left-handed coordinate system. Additionally, manual intervention is only required once. When we use other dataset like Surreal or 3DHP, we do not need to define these mapping equations any more. The mapping equations of Human3.6M are also suitable for other datasets. The corresponding mapping equations are shown as follows:

(1). Right Hip

$$\begin{cases} [x_p,\ y_p,\ z_p] = [\dfrac{l_{hp\_rh}}{|l_{hp\_rh}|},\ x_p \otimes z_p,\ \dfrac{l_{hp\_rh} \otimes l_{hp\_s}}{|l_{hp\_rh} \otimes l_{hp\_s}|}] \\ [x_c,\ y_c,\ z_c] = [\dfrac{l_{rk\_ra} \otimes l_{rh\_rk}}{|l_{rk\_ra} \otimes l_{rh\_rk}|},\ \dfrac{l_{rh\_rk}}{|l_{rh\_rk}|},\ y_c \otimes x_c] \end{cases} \tag{1}$$

(2). Right Knee

$$\begin{cases} [x_p,\ y_p,\ z_p] = [\dfrac{l_{rk\_ra} \otimes l_{rh\_rk}}{|l_{rk\_ra} \otimes l_{rh\_rk}|},\ \dfrac{l_{rh\_rk}}{|l_{rh\_rk}|},\ y_p \otimes x_p] \\ [x_c,\ y_c,\ z_c] = [x_p,\ \dfrac{l_{rk\_ra}}{|l_{rk\_ra}|},\ y_c \otimes x_c] \end{cases} \tag{2}$$

(3). Left Hip

$$\begin{cases} [x_p,\ y_p,\ z_p] = [\dfrac{-l_{hp\_lh}}{|l_{hp\_lh}|},\ x_p \otimes z_p,\ \dfrac{-l_{hp\_lh} \otimes l_{hp\_s}}{|-l_{hp\_lh} \otimes l_{hp\_s}|}] \\ [x_c,\ y_c,\ z_c] = [\dfrac{l_{lk\_la} \otimes l_{lh\_lk}}{|l_{lk\_la} \otimes l_{lh\_lk}|},\ \dfrac{l_{lh\_lk}}{|l_{lh\_lk}|},\ y_c \otimes x_c] \end{cases} \tag{3}$$

(4). Left Knee

$$\begin{cases} [x_p,\ y_p,\ z_p] = [\dfrac{l_{lk\_la} \otimes l_{lh\_lk}}{|l_{lk\_la} \otimes l_{lh\_lk}|},\ \dfrac{l_{lh\_lk}}{|l_{lh\_lk}|},\ y_p \otimes x_p] \\ [x_c,\ y_c,\ z_c] = [x_p,\ \dfrac{l_{lk\_la}}{|l_{lk\_la}|},\ y_c \otimes x_c] \end{cases} \tag{4}$$

(5). Right Shoulder

$$\begin{cases} [x_p,\ y_p,\ z_p] = [\dfrac{l_{n\_rs}}{|l_{n\_rs}|},\ x_p \otimes z_p,\ \dfrac{-l_{s\_n} \otimes l_{n\_rs}}{|-l_{s\_n} \otimes l_{n\_rs}|}] \\ [x_c,\ y_c,\ z_c] = [\dfrac{l_{rs\_re}}{|l_{rs\_re}|},\ \dfrac{l_{re\_rw} \otimes l_{rs\_re}}{|l_{re\_rw} \otimes l_{rs\_re}|},\ y_c \otimes x_c] \end{cases} \tag{5}$$

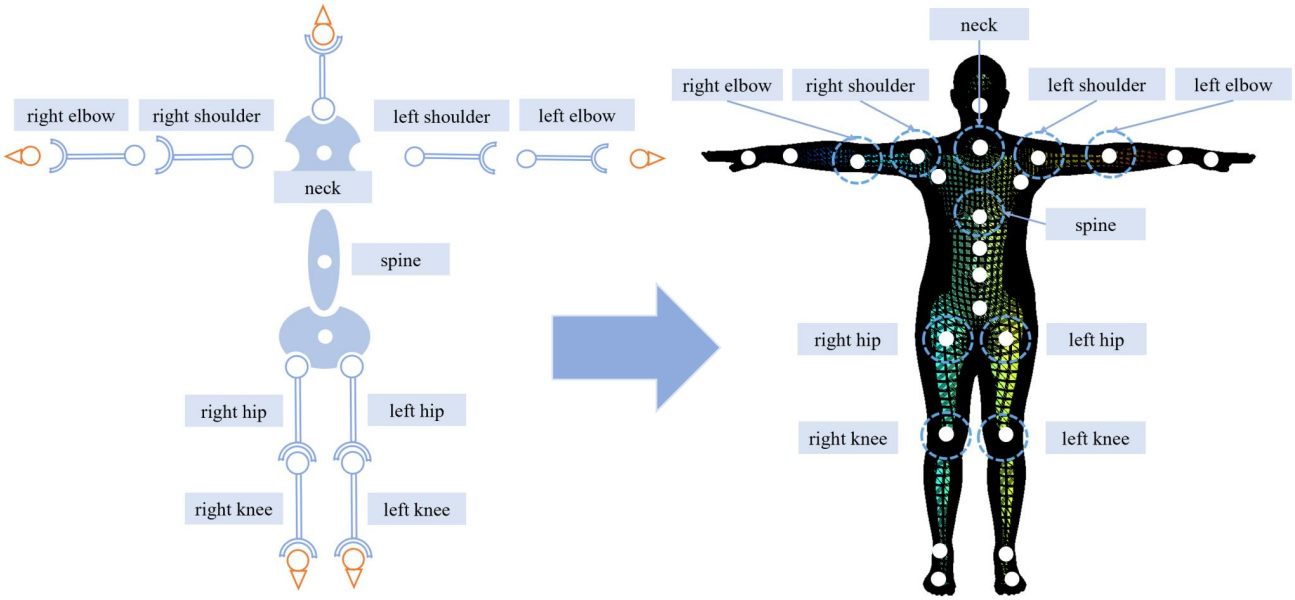

Figure 3: Detailed information of ten nine units (skeleton in the left, and SMPL model in the right), including right hip, right knee, left hip, left knee, right shoulder, right elbow, left shoulder, left elbow, neck, and spine.

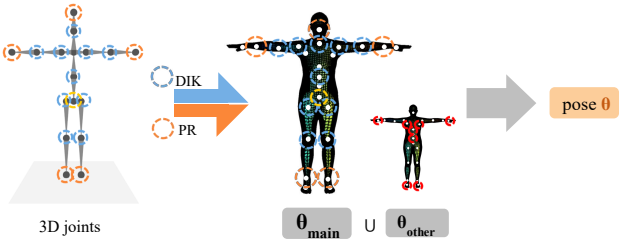

Figure 4: Detailed correspondence between 3D skeleton and 3D rotation. Blue and orange circle denotes joints matched by DIK and PR modules, respectively. And the red circle indicates that no suitable joints to match. Notably, yellow circle representing root orientation is trained by the network independently. $\boldsymbol{\theta}_{main}$ indicates the collection of five local 3D rotations (orange circles), ten 3D local rotations (blue circles), one global 3D rotation (yellow circles) in SMPL, and $\boldsymbol{\theta}_{other}$ indicates the other local 3D rotations.

(6). Right Elbow

$$\begin{cases} [x_p,\ y_p,\ z_p] = [\dfrac{l_{rs\_re}}{|l_{rs\_re}|},\ \dfrac{l_{re\_rw} \otimes l_{rs\_re}}{|l_{re\_rw} \otimes l_{rs\_re}|},\ y_p \otimes x_p] \\ [x_c,\ y_c,\ z_c] = [\dfrac{l_{re\_rw}}{|l_{re\_rw}|},\ y_p,\ y_c \otimes x_c] \end{cases} \tag{6}$$

(7). Left Shoulder

$$\begin{cases} [x_p,\ y_p,\ z_p] = [\dfrac{-l_{n\_ls}}{|l_{n\_ls}|},\ x_p \otimes z_p,\ \dfrac{l_{n\_s} \otimes -l_{n\_ls}}{|l_{n\_s} \otimes -l_{n\_ls}|}] \\ [x_c,\ y_c,\ z_c] = [\dfrac{-l_{ls\_le}}{|l_{ls\_le}|},\ \dfrac{-l_{ls\_le} \otimes -l_{le\_lw}}{|-l_{ls\_le} \otimes -l_{le\_lw}|},\ y_c \otimes x_c] \end{cases} \tag{7}$$

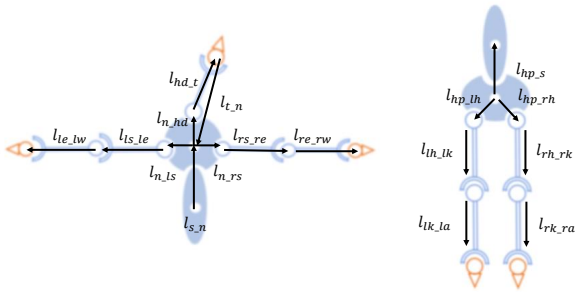

Figure 5: Matching details of ten units, including right hip, right knee, left hip, left knee, right shoulder, right elbow, left shoulder, left elbow, neck, and spine.

(8). Left Elbow

$$\begin{cases} [x_p,\ y_p,\ z_p] = [\dfrac{-l_{ls\_le}}{|l_{ls\_le}|},\ \dfrac{l_{ls\_le} \otimes l_{le\_lw}}{|l_{ls\_le} \otimes l_{le\_lw}|},\ y_p \otimes x_p] \\ [x_c,\ y_c,\ z_c] = [\dfrac{-l_{le\_lw}}{|l_{le\_lw}|},\ y_p,\ y_c \otimes x_c] \end{cases} \tag{8}$$

(9). Neck

$$\begin{cases} [x_p,\ y_p,\ z_p] = [z_p \otimes y_p,\ \dfrac{-l_{s\_n}}{|l_{s\_n}|},\ \dfrac{-l_{s\_n} \otimes l_{n\_rs}}{|-l_{s\_n} \otimes l_{n\_rs}|}] \\ [x_c,\ y_c,\ z_c] = [\dfrac{l_{hd\_t} \otimes -l_{n\_hd}}{|l_{hd\_t} \otimes -l_{n\_hd}|},\ \dfrac{l_{t\_n}}{|l_{t\_n}|},\ y_c \otimes x_c] \end{cases} \tag{9}$$

Method	HMR	GraphCMR	SPIN	Pose2Mesh	ours
PMPJPE	88.4	104.6	55.6	48.7	**46.9**

Table 2: The accuracy comparison between state-of-the-art methods and ours on Human3.6M under the same fully supervised setting.

Trainset	Human3.6M	+3DHP	+3DHP+3DPW
PMPJPE	100.0	96.9	90.3

Table 3: Analysis on the generalization ability of proposed framework. Performance is given on the test set of 3DPW using different training datasets.

(10). Spine

$$\begin{cases} [x_p,\ y_p,\ z_p] = [z_p \otimes y_p,\ \dfrac{-l_{hp\_s}}{|l_{hp\_s}|},\ \dfrac{-l_{hp\_s} \otimes l_{hp\_rh}}{|-l_{hp\_s} \otimes l_{hp\_rh}|}] \\ [x_c,\ y_c,\ z_c] = [z_c \otimes y_c,\ \dfrac{-l_{s\_n}}{|l_{s\_n}|},\ \dfrac{-l_{s\_n} \otimes l_{n\_rs}}{|-l_{s\_n} \otimes l_{n\_rs}|}] \end{cases} \tag{10}$$

3. Further Experiments & Ablation Study

To perform a more comprehensive and convincing evaluation on proposed framework, we conduct some supplementary experiments: *performance in fully supervised setting*, *generalization ability given more in-the-wild images* and *usage of DIK module as a supervision method*.

With Full Access To 3D Annotation. To further illustrate the effectiveness of proposed framework and compare against state-of-the-art methods, we extend our framework to fully supervised setting. We add a loss term $\mathcal{L} = ||\hat{\mathbf{J}}^{3D} - \mathbf{J}^{3D}_{gt}||^2_2$ to supervise our lifting module by GT 3D skeleton $\mathbf{J}^{3D}_{gt}$. As shown in Tab. 2, PMPJPE improves by a large margin compared with Pose2Mesh [3] (*48.7* $\rightarrow$ *46.9*), which obviously outperforms other methods.

Generalization Ability of Proposed Framework. To further prove that our method can generalize better with more in-the-wild training data, we provide additional experiments results in Tab. 3. We train on different combinations of 3D datasets and test on 3DPW. When introduce images from MPI-INF-3DHP and 3DPW into training, PMPJPE decreases from 100 to 96.9, and further to 90.3, proving the generalization ability of proposed framework.

Usage of The DIK Module. In proposed framework, the DIK module is one stage in the pipeline of forward pass, deriving rotation vectors from upstream 3D keypoint locations. DIK module is efficient and can be plugged into any paradigm. However, we find that such practice may fail to match well with valid pose space of SMPL and suffer from temporal inconsistency. To alleviate this problem, we provide a more robust solution: *use the DIK module as supervision*. To be specific, we use several residual blocks to learn rotations vectors from upstream 3D skeletons. In this way, deep neural networks are encouraged to find a more valid manifold for rotation vectors. Preliminary experiments show that PVE decreases from *120.8* to *115.4* in this setting. Detailed experiments and analysis are left for future work.

4. Running Time

On a Titan X GPU, it takes 60ms to estimate SMPL parameters in our framework from a monocular RGB image. If 2D keypoints and segmentation are obtained beforehand, it will only take 10ms for a single forward pass of our pose matching branch and shape matching branch. This is much faster than optimization-based methods. For example, SMPLify [1] reports roughly 1 minute for the optimization. Moreover, our network is competitive to other learning-based methods in terms of speed (e.g. Pavlakos et al. [6] report 50ms per image), but our performance surpasses theirs by significant margins.

5. Visualization Results

We present the visual results on Human3.6M, MPI-INF-3DHP, Surreal, respectively. More qualitative and quantitative results can be seen in our project page [1].

6. Potential Applications in the Future

Our Skeleton2Mesh, a novel lightweight framework which relies on a minimal set of kinematics prior knowledge, can apply to real-time task. We provide two demos (i.e., skeleton to SMPL model with mean shape, and skeleton to robot arm) in our project page, and the corresponding introductions respectively. In the future, we would like to extend such kind of framework for real-time robot or carton character control in virtual reality or other applications.

References

[1] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter V. Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: automatic estimation of 3d human pose and shape from a single image. In *ECCV*, pages 561–578, 2016. 4

[2] Ching-Hang Chen, Ambrish Tyagi, Amit Agrawal, Dylan Drover, Rohith MV, Stefan Stojanov, and James M. Rehg. Unsupervised 3d pose estimation with geometric self-supervision. In *CVPR*, pages 5714–5724. Computer Vision Foundation / IEEE, 2019. 1

[3] Hongsuk Choi, Gyeongsik Moon, and Kyoung Mu Lee. Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose.

[1] https://sites.google.com/view/skeleton2mesh

In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VII*, volume 12352 of *Lecture Notes in Computer Science*, pages 769–787. Springer, 2020. 4

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. 1

[5] Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. A simple yet effective baseline for 3d human pose estimation. In *ICCV*, pages 2659–2668, 2017. 1

[6] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *CVPR*, pages 459–468, 2018. 4