Supplementary Materials: Towards Alleviating the Modeling Ambiguity of Unsupervised Monocular 3D Human Pose Estimation

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

1. Comparison with SOTA Methods

As illustrated in Tab. 1, the information available for unsupervised 3D pose estimation largely varies in previous work [2, 5]. To more comprehensively evaluate our model, we compare our method with these mentioned works. Specifically, Chen et.al. [2] transfer the 2d skeletons from other datasets to the domain of Human3.6M, which requires large-scale source data for such kind of adaptation. Li et.al. [5] show that a small portion of the labelled 3d skeletons is sufficient to train a 3D pose estimation network with promising performance, which, however, is still a high cost to acquire in many applications. In contrast to their designs, we only use 2d skeletons, which is easy to extract from monocular images, and achieve satisfying results. More qualitative and quantitative results can be seen in our project page [1].

Algorithm	2D Pose	Unpaired 3D Pose	Paired 3D Pose	3D Prior	PMPJPE
Chen et al.[1]	✓	✗	✗	✗	68.0
Li et al.[5]	✗	✗	✓	✗	66.5
Kundu et al. [4]	✓	✗	✗	✓	63.8
Kundu et al. [3]	✓	✓	✗	✗	62.4
Ours	✓	✗	✗	✗	**52.3**

Table 1: Comparison of our method against previous works, in terms of supervision signals.

2. Detailed Architecture

In Sec. 3.4 we mention that 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 [6].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, we use five residual blocks for the lifting network Φ_G, one residual block for the scale network and two residual 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 and we denote these residual blocks as "n $\times$ Res Block".

Geometric Random Rotation. Geometric random rotation in our paper is similar to [1]. 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 $\mathbf{R}$.

Analysis on Essential Operations. Without access to the source code of [1], we re-implement the baseline used in [1]. As shown in Tab. 2, the same baseline based on our implementation is better than that in [1]. We attribute this improvement to the tuning of *essential operations*(e.g. adversarial training and random rotation), To be specific, 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 [1]) results in better performance. Rotations around x and z axes are limited to $\pi/9$ and $\pi/18$ respectively. Similarly, dataset-specific tuning of *essential operations* can also boost the performance on MPI-INF-3DHP dataset. We show the results in Tab. 3. More details in our implementation can be seen in our re-

*corresponding author
[1]https://sites.google.com/view/ambiguity-aware-hpe

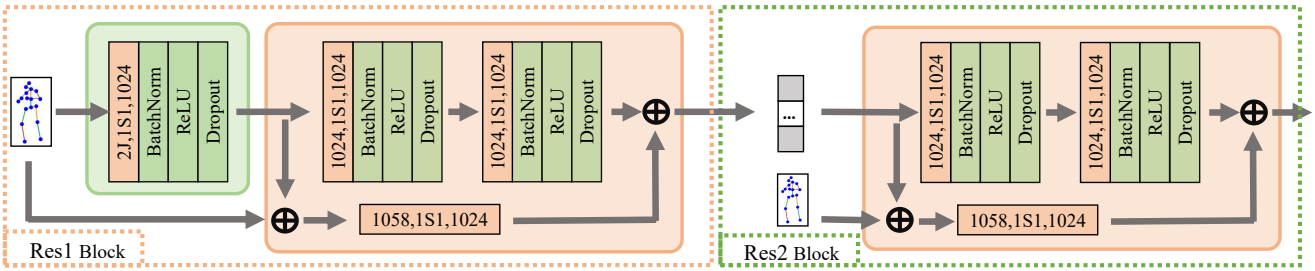

Figure 1: The detailed architecture of the Residual Block including Res1 Block and Res2 Block. The input consists of 2D keypoints with J=17 joints. Convolutional layers are in red where $2J$, $1S1$, 1024 denote $2J$ input channels, kernels of size 1 with stride 1, and 1024 output channels. And the skip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

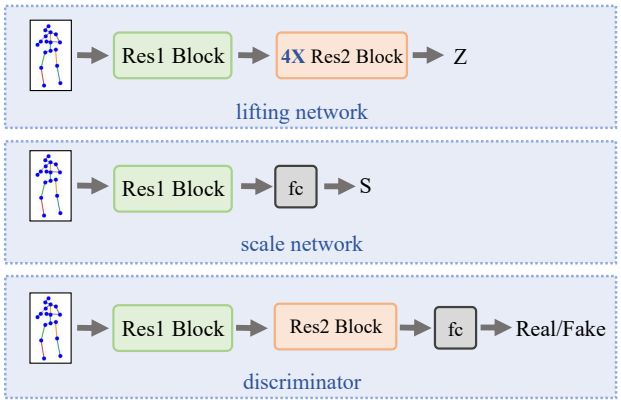

Figure 2: The detailed architecture of the lifting network, the scale network and the discriminator. fc denotes fully connected layer with 1024 dimension features

Baselines	Adv. Loss	Azimuth	MPJPE	PMPJPE
Baseline[1]	BCE	π	-	58.0
*Baseline[1]	BCE	π	108.6	54.5
Variant	LS	π	110.0	50.9
Final Baseline	LS	$7/9 * \pi$	**105.0**	**46.0**

Table 2: Analysis on essential operations on the baseline for Human3.6M. * indicates our implementation according to [1]. Based on our implementation, we further make some improvements, as listed in the table. **Adv Loss** indicates adversarial loss function. **BCE** indicates binary cross entropy loss function. **LS** indicates least square loss function. **Azimuth** indicates the maximum azimuth in geometric random rotation.

leased code.

Adv. Loss	#Critics	Azimuth	3D PCK	AUC
Binary Cross Entropy	3	π	84.1	48.9
Least Square	3	π	85.1	50.4
Least Square	2	π	85.7	51.2
Least Square	2	$7\pi/9$	86.1	51.6
Least Square	2	$8\pi/9$	**86.2**	**51.7**

Table 3: Analysis on essential operations on MPI-INF-3DHP. **Adv Loss** indicates the adversarial loss function. **#Critics** represents how many iterations the discriminator is trained with when lifting module is trained with one iteration. **Azimuth** denotes the maximum azimuth angle involved in random rotation.

3. Another View on Scale Module

Since $D \gg d_i$ and $Z_i = D + d_i$, we have $Z_i \approx D$, where the perspective projection can be approximated as follows:

$$x_i = \frac{X_i}{D} \cdot f_x. \tag{1}$$

We can see that x_i is proportional to $\frac{X_i}{D}$ (i.e., $x_i \sim \frac{X_i}{D}$). Scale estimation module is firstly utilized to infer the scale of 3D pose S_{3D}, i.e., $x_i = \frac{S_{3D} \cdot X_i}{D}$. It can be alternatively written as $x_i = \frac{X_i}{S_D \cdot D}$, where $S_D = \frac{1}{S_{3D}}$. This means the scale estimation module not only restricts the scale of estimated 3D skeleton, but also is able to estimate the relative depth of the corresponding estimated 3D skeleton. Please refer to our project page for visually appealing results.

4. More Visual Results

In the following paragraphs, we also report the detailed visual results about the effectiveness of temporal scale consistency and multi-view motion consistency. Then we present the visual results of 3D pose estimation based on our method on Human3.6M, MPI-INF-3DHP, Surreal, respectively. More video results can be seen in our project page.

Visual results of temporal scale consistency. As illustrated in Fig. 5, two sequences in Human3.6M predicted by our proposed method with/without temporal scale consistency are presented in terms of MPJPE. To demonstrate the effectiveness of temporal scale consistency, we visualize predicted 3D skeletons and the ground truth in the same coordinate system simultaneously. Furthermore, compared to the baseline, we can observe that adding temporal scale consistency is able to achieve more plausible and valid results.

Visual results of multi-view motion consistency. As shown in Fig. 4, the sequence on MPI-INF-3DHP is exhibited with two extra views, which proves the effectiveness of multi-view motion consistency. Experimental results show that our method is much more effective than [1]. We have released our complete code including data processing, visualization, evaluation, etc. in our supplementary materials.

References

[1] 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, 2019. 1, 2, 3

[2] Xipeng Chen, Kwan-Yee Lin, Wentao Liu, Chen Qian, and Liang Lin. Weakly-supervised discovery of geometry-aware representation for 3d human pose estimation. In *CVPR*, pages 10895–10904, 2019. 1

[3] Jogendra Nath Kundu, Siddharth Seth, Varun Jampani, Mugalodi Rakesh, R. Venkatesh Babu, and Anirban Chakraborty. Self-supervised 3d human pose estimation via part guided novel image synthesis. In *CVPR*, pages 6151–6161, 2020. 1

[4] Jogendra Nath Kundu, Siddharth Seth, Rahul M. V., Mugalodi Rakesh, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Kinematic-structure-preserved representation for unsupervised 3d human pose estimation. In *AAAI*, pages 11312–11319, 2020. 1

[5] Zhi Li, Xuan Wang, Fei Wang, and Peilin Jiang. On boosting single-frame 3d human pose estimation via monocular videos. In *ICCV*, pages 2192–2201, 2019. 1

[6] 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

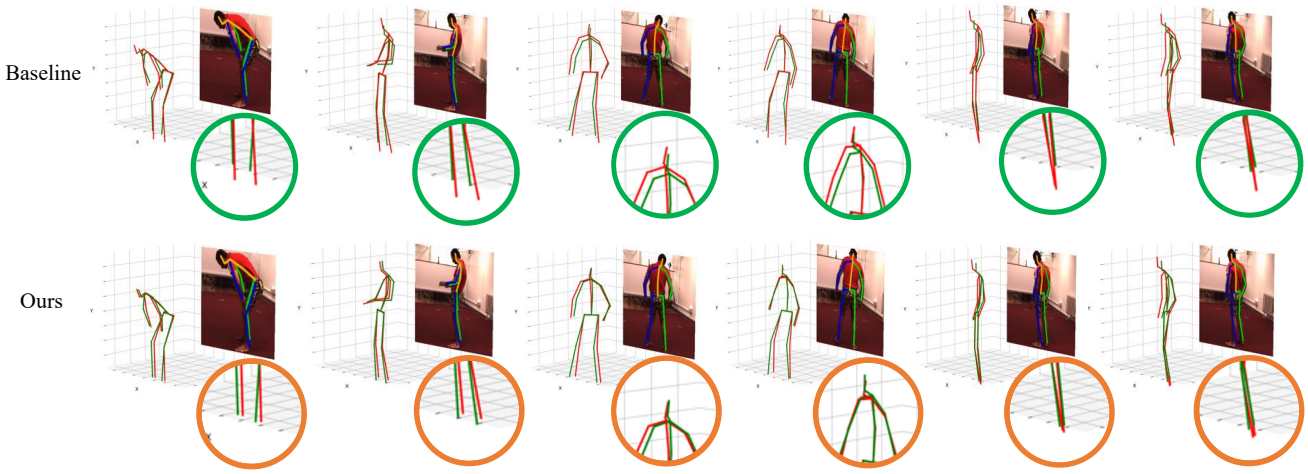

Figure 3: Qualitative results on Human3.6M. **Top row:** baseline predictions(red) along with ground truth(green). **Bottom row:** our predicted predictions(red) along with ground truth(green).

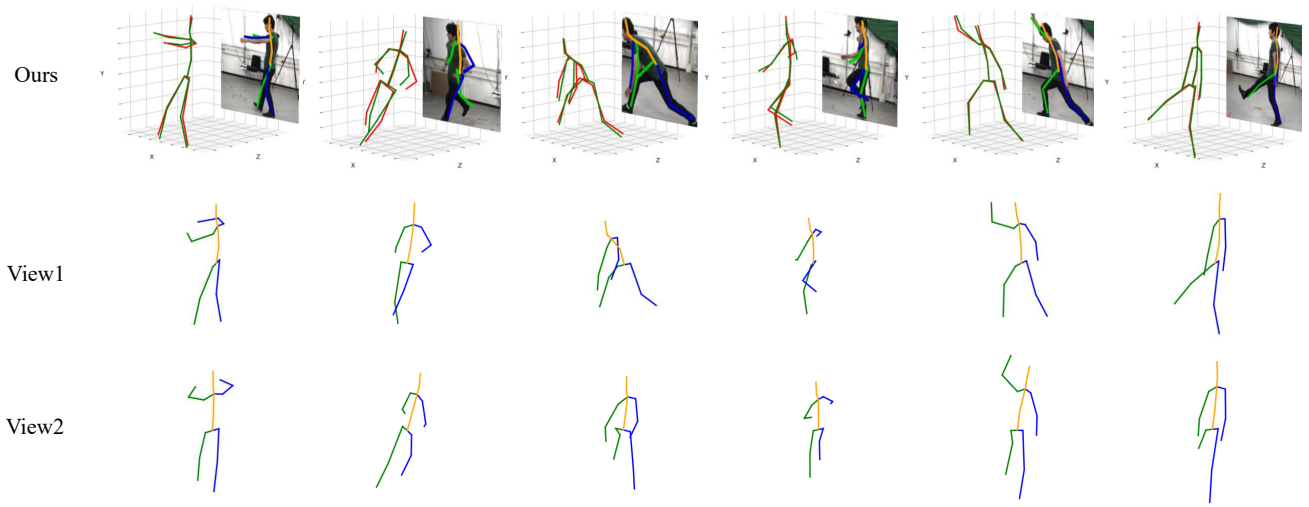

Figure 4: Qualitative results on MPI-INF-3DHP. **Top row:** Our predicted predictions (red) along with ground truth. **Middle row:** One view of our predicted predictions(rotated around y axis by $\pi/2$). **Bottom row:** Another view of our predicted predictions(rotated around y axis by $3\pi/4$).

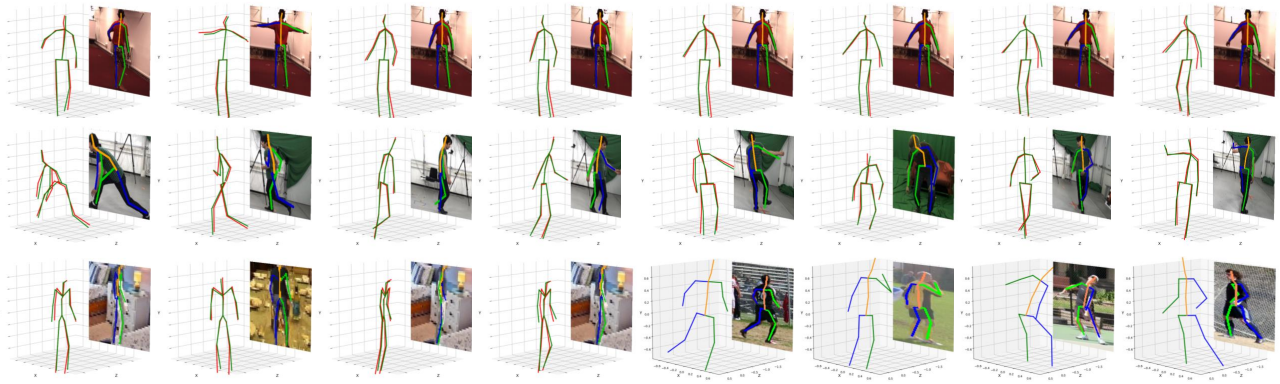

Figure 5: Qualitative results on Human3.6M, MPI-INF-3DHP, Surreal, and LSP. **Top Row:**Human3.6M dataset. **Middle Row:** MPI-INF-3DHP dataset. **Bottom Row:** Surreal and LSP datasets.