Di^{2}Pose: Discrete Diffusion Model for Occluded 3D Human Pose Estimation **Appendix**

Weiquan Wang[1], Jun Xiao[1], Chunping Wang[2], Wei Liu[3], Zhao Wang[1], Long Chen[4]
[1]Zhejiang University [2]Finvolution Group [3]Tencent
[4]Hong Kong University of Science and Technology
{wqwangcs, junx}@zju.edu.cn, wangchunping02@xinye.com,
wl2223@columbia.edu, zhao_wang@zju.edu.cn, longchen@ust.hk

Summary

A Preliminary: Continuous Diffusion Model   **1**

B Mathematical Proofs   **2**

C Algorithms for Discrete Diffusion Process   **3**

    C.1 Training Procedure 3

    C.2 Inference Procedure 3

D Additional Implementation Details   **4**

E Additional Experimental Results   **4**

    E.1 Quantitative Results 4

    E.2 Qualitative Results 5

F Broader Impacts   **5**

In this Appendix, we provide relevant preliminary knowledge, mathematical proofs, complete training and inference algorithms, additional experimental results, more implementation details about our Di^{2}Pose and limitations and broader impacts.

A Preliminary: Continuous Diffusion Model

The continuous diffusion model consists of two primary processes: the *forward process* and the *reverse process*. The forward process methodically corrupts the original data x_0 into a noisy latent variable x_S, which converges to a stationary distribution (e.g., a Gaussian distribution). Conversely, the reverse process aims to reconstruct the original data x_0 from x_S, utilizing learned parameters.

Forward Process Starting with x_0 drawn from the distribution $q(x_0)$, the forward process incrementally corrupts x_0 through a sequence of latent variables $x_{1:S} = (x_1, x_2, \ldots, x_S)$, where each x_s

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

retains the same dimensionality as $\boldsymbol{x}_0$. This transformation is modeled as a fixed Markov chain:

$$q(\boldsymbol{x}_{1:S}|\boldsymbol{x}_0) = \prod_{s=1}^{S} q(\boldsymbol{x}_s|\boldsymbol{x}_{s-1}). \tag{1}$$

where each transition $q(\boldsymbol{x}_s|\boldsymbol{x}_{s-1})$ is defined by a Gaussian distribution:

$$q(\boldsymbol{x}_s|\boldsymbol{x}_{s-1}) = \mathcal{N}(\boldsymbol{x}_s; \sqrt{1-\eta_s}\boldsymbol{x}_{s-1}, \eta_s\boldsymbol{I}) \tag{2}$$

Here, η_s is a small positive constant that follows a predefined schedule $(\eta_1, \eta_2, \ldots, \eta_S)$, allowing the data to progressively approach an isotropic Gaussian distribution, $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, as s increases. The overall transition from $\boldsymbol{x}_0$ to $\boldsymbol{x}_s$ can thus be expressed as:

$$q(\boldsymbol{x}_s|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_s; \sqrt{\overline{\zeta}_s}\boldsymbol{x}_0, (1-\overline{\zeta}_s)\boldsymbol{I}) \tag{3}$$

where $\zeta_s = 1 - \eta_s$ and $\overline{\zeta}_s = \prod_{i=1}^{s}\zeta_i$.

Reverse Process In the reverse process, the model aims to convert the latent variable $\boldsymbol{x}_S$, which is assumed to follow the distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, back into the original data $\boldsymbol{x}_0$. The joint probability distribution is given by:

$$p_\theta(\boldsymbol{x}_{0:S}) = p(\boldsymbol{x}_S) \prod_{s=1}^{S} p_\theta(\boldsymbol{x}_{s-1}|\boldsymbol{x}_s) \tag{4}$$

The conditional distributions involved are inferred using Bayes rule as follows:

$$q(\boldsymbol{x}_{s-1}|\boldsymbol{x}_s, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_s|\boldsymbol{x}_{s-1}, \boldsymbol{x}_0)q(\boldsymbol{x}_{s-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_s|\boldsymbol{x}_0)} \tag{5}$$

To optimize the generative model $p_\theta(\boldsymbol{x}_0)$ for fitting the data distribution $q(\boldsymbol{x}_0)$, we minimize a variational upper bound on the negative log-likelihood:

$$\mathcal{L}_{vb} = \mathbb{E}_{q(\boldsymbol{x}_0)}\Big[D_{KL}[q(\boldsymbol{x}_S|\boldsymbol{x}_0)||p(\boldsymbol{x}_S)] + \sum_{s=1}^{S} \mathbb{E}_{q(\boldsymbol{x}_s|\boldsymbol{x}_0)}\big[D_{KL}[q(\boldsymbol{x}_{s-1}|\boldsymbol{x}_s, \boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{s-1}|\boldsymbol{x}_s)]\big]\Big]. \tag{6}$$

However, continuous diffusion models are not applicable in discrete spaces, such as quantized token indices $\mathbf{k} = (k_1, k_2, \ldots, k_N)$ where each k_i assumes one of $|\mathcal{C}|$ discrete values. This limitation arises because Gaussian noise cannot corrupt discrete elements in a meaningful way. Thus, modeling in discrete spaces necessitates the development of discrete diffusion processes.

B Mathematical Proofs

In this section, we provide a detailed mathematical proofs for Eq. (6), which can quickly calculate $q(\mathbf{k}_s|\mathbf{k}_0)$ according to Eq. (2).

Concretely, we use mathematical induction to prove Eq. (6). At first, we have following conditional information:

$$\alpha_s, \beta_s \in [0, 1], \alpha_s = 1 - |\mathcal{C}|\beta_s - \gamma_s,$$
$$\overline{\alpha}_s = \prod_{i=1}^{s}\alpha_s, \overline{\gamma}_s = 1 - \prod_{i=1}^{s}(1-\gamma_s), \overline{\beta}_s = (1 - \overline{\alpha}_s - \overline{\gamma}_s)/|\mathcal{C}|. \tag{7}$$

Now we want to prove that $\overline{\mathbf{M}}_s\boldsymbol{c}(\mathbf{k}_0) = \overline{\alpha}_s\boldsymbol{c}(\mathbf{k}_0) + (\overline{\gamma}_s - \overline{\beta}_s)\boldsymbol{c}(|\mathcal{C}| + 1) + \overline{\beta}_s$. Firstly, when $s = 1$, we have:

$$\overline{\mathbf{M}}_1\boldsymbol{c}(\mathbf{k}_0) = \begin{cases} \overline{\alpha}_1 + \overline{\beta}_1, & \mathbf{k} = \mathbf{k}_0 \\ \overline{\beta}_1, & \mathbf{k} \neq \mathbf{k}_0 \text{ and } \mathbf{k} \neq |\mathcal{C}| + 1 \\ \overline{\gamma}_1, & \mathbf{k} = |\mathcal{C}| + 1 \end{cases} \tag{8}$$

which is clearly hold. Suppose the Eq. (6) holds at step s, then for $s = s + 1$, we have:

$$\overline{\mathbf{M}}_{s+1}\boldsymbol{c}(\mathbf{k}_0) = \mathbf{M}_{\mathbf{k}+1}\overline{\mathbf{M}}_t\boldsymbol{c}(\mathbf{k}_0). \tag{9}$$

Now we consider three conditions:
(1) when $\mathbf{k} = \mathbf{k}_0$ in step $s + 1$, we have:

2

$$
\begin{aligned}
\mathbf{M}_{s+1}\boldsymbol{c}(\mathbf{k}_0)_{(\mathbf{k})} &= \overline{\beta}_s\beta_{s+1}(|\mathcal{C}|-1) + (\alpha_{s+1}+\beta_{s+1})(\overline{\alpha}_s+\overline{\beta}_s) \\
&= \overline{\beta}_s(|\mathcal{C}|\beta_{s+1}+\alpha_{s+1}) + \overline{\alpha}_s(\alpha_{s+1}+\beta_{s+1}) \\
&= \frac{1}{|\mathcal{C}|}(\overline{\beta}_s(1-\gamma_{s+1}) + \overline{\alpha}_s\beta_{s+1} - \overline{\beta}_{s+1})*|\mathcal{C}| + \overline{\alpha}_{s+1} + \overline{\beta}_{s+1} \\
&= \frac{1}{|\mathcal{C}|}[(1-\overline{\alpha}_s-\overline{\gamma}_s)(1-\gamma_{s+1}) + |\mathcal{C}|\overline{\alpha}_s\beta_{s+1} - (1-\overline{\alpha}_{s+1}-\overline{\gamma}_{s+1})] + \overline{\alpha}_{s+1} + \overline{\beta}_{s+1} \\
&= \frac{1}{|\mathcal{C}|}[(1-\overline{\gamma}_{s+1}) - \overline{\alpha}_s(1-\gamma_{s+1}-K\beta_{s+1}) - (1-\overline{\gamma}_{s+1}) + \overline{\alpha}_{s+1}] + \overline{\alpha}_{s+1} + \overline{\beta}_{s+1} \\
&= \overline{\alpha}_{s+1} + \overline{\beta}_{s+1}.
\end{aligned}
\tag{10}
$$

(2) when $\mathbf{k} = |\mathcal{C}| + 1$ in step $s+1$, we have:

$$
\mathbf{M}_{s+1}\boldsymbol{c}(\mathbf{k}_0)_{(\mathbf{k})} = \overline{\gamma}_s + (1-\overline{\gamma}_s)\gamma_{s+1} = 1 - (1-\overline{\gamma}_{s+1}) = \overline{\gamma}_{s+1}.
\tag{11}
$$

(3) when $\mathbf{k} \neq \mathbf{k}_0$ and $\mathbf{k} \neq |\mathcal{C}| + 1$ in step $s+1$, we have:

$$
\begin{aligned}
\mathbf{M}_{s+1}\boldsymbol{c}(\mathbf{k}_0)_{(\mathbf{k})} &= \overline{\beta}_s(\alpha_{s+1}+\beta_{s+1}) + \overline{\beta}_s\beta_{s+1}(|\mathcal{C}|-1) + \overline{\alpha}_s\beta_{s+1} \\
&= \overline{\beta}_s(\alpha_{s+1}+|\mathcal{C}|\beta_{s+1}) + \overline{\alpha}_s\beta_{s+1} \\
&= \frac{1-\overline{\alpha}_s-\overline{\gamma}_s}{|\mathcal{C}|}*(1-\gamma_{s+1}) + \overline{\alpha}_s\beta_{s+1} \\
&= \frac{1}{|\mathcal{C}|}(1-\overline{\gamma}_{s+1}) + \overline{\alpha}_s(\beta_{s+1} - \frac{1-\gamma_{s+1}}{|\mathcal{C}|}) \\
&= \overline{\beta}_{s+1}.
\end{aligned}
\tag{12}
$$

The proof of Eq. (6) is completed. Notably, according to Eq. (6), the computation cost of $q(\mathbf{k}_s|\mathbf{k}_0)$ can be reduced from $O(|\mathcal{C}|^2 S)$ to $O(|\mathcal{C}|)$.

C Algorithms for Discrete Diffusion Process

In this section, we provide complete training and inference algorithms for discrete diffusion process.

C.1 Training Procedure

The discrete diffusion process aims to model quantized 3D pose tokens in a discrete space. This involves utilizing a 2D image I and its corresponding 3D human pose $\mathbf{P}$ as inputs. The image I serves as a contextual condition, while $\mathbf{P}$ is converted into discrete tokens for modeling.

Firstly, the 3D human pose $\mathbf{P}$ is encoded by $f_{PE}(\cdot)$ and subsequently quantized using the FSQ technique, resulting in multiple discrete tokens. Concurrently, a pre-trained Image Encoder extracts contextual features from I, producing a conditional feature sequence $\boldsymbol{y}$. During the forward process, we sample s from a uniform distribution $\{1, 2, ..., S-1, S\}$ and compute $q(\mathbf{k}_s|\mathbf{k}_0)$ based on Eq. (6). In the reverse process, the pose denoiser $f_\theta(\mathbf{k}_{s-1}|\mathbf{k}_s, \boldsymbol{y})$ is trained to estimate $q(\mathbf{k}_{s-1}|\mathbf{k}_s, \mathbf{k}_0)$. Finally, the overall loss is calculated according to Eq. (10), and the parameters of the pose denoiser θ are updated accordingly.

The complete training algorithm for the discrete diffusion process is presented in Algorithm 1.

C.2 Inference Procedure

In the inference process, our objective is to recover the 3D human pose $\hat{\mathbf{P}}$ from an input 2D image and discrete tokens.

Initially, all pose tokens are either masked or initialized randomly, which is achieved by sampling from the stationary distribution $p(\mathbf{k}_S)$. The 2D image I is encoded using the pre-trained Image Encoder. Subsequently, we predict $f_\theta(\mathbf{k}_{s-1}|\mathbf{k}_s, \boldsymbol{y})$ step by step until the pose tokens are fully

3

Algorithm 1 Training Algorithm for the discrete diffusion process.

Require:
 A transition matrix $\mathbf{M}_s$, the number of steps S, parameters of pose denoiser θ, training epoch T, pose dataset $\boldsymbol{D}$ (including 2D image I and 3D human pose $\mathbf{P}$), and the well-learned pose encoder $f_{PE}(\cdot)$.
 1: **for** $i = 1$ to T **do**
 2:   **for** $(I, \mathbf{P})$ in $\boldsymbol{D}$ **do**
 3:     $\mathbf{k}_0 = \text{FSQ}(f_{PE}(\mathbf{P}))$, $\boldsymbol{y} = \text{ImageEncoder}(I)$;
 4:     sample s from $\text{Uniform}\{1, 2, ..., S-1, S\}$;
 5:     calculate $q(\mathbf{k}_s | \mathbf{k}_0)$ based on Eq. (6);
 6:     estimate $f_\theta(\mathbf{k}_{s-1} | \mathbf{k}_s, \boldsymbol{y})$;
 7:     calculate loss according to Eq. (10);
 8:     update θ;
 9:   **end for**
10: **end for**
11: **return** θ.

Algorithm 2 Inference Algorithm for the discrete diffusion process.

Require:
 The number of steps S, input 2D image I, the pose decoder $f_{PD}(\cdot)$, parameters of pose denoiser θ, stationary distribution $p(\mathbf{k}_S)$;
 1: $s = S$, $\boldsymbol{y} = \text{ImageEncoder}(I)$;
 2: sample $\mathbf{k}_s$ from $p(\mathbf{k}_S)$;
 3: **while** $s > 0$ **do**
 4:   $\mathbf{k}_s \leftarrow$ sample from $p_\theta(\mathbf{k}_{s-1} | \mathbf{k}_s, \boldsymbol{y})$
 5:   $s \leftarrow (s - 1)$
 6: **end while**
 7: **return** $f_{PD}(\mathbf{k}_s)$.
---

recovered. Finally, the reconstructed tokens are decoded using the pose decoder $f_{PE}(\cdot)$, yielding the recovered 3D pose $\hat{\mathbf{P}}$.

The complete inference algorithm for the discrete diffusion process is presented in Algorithm 2.

D   Additional Implementation Details

All experiments are carried out on one NVIDIA A100 PCIe GPU. The proposed Di^{2}Pose is completely implemented in PyTorch [6]. In this section, we provide the detailed training settings for the pose quantization step and the discrete diffusion process.

For the pose quantization step, we employ the AdamW [4] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, adhering to a base learning rate of 1e-3 and a weight decay parameter of 0.15. The training process is configured with a batch size of 256 across a total of 20 epochs.

For the discrete diffusion process, we still utilize the the AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.96$, adhering to a base learning rate of 5.5e-4 and a weight decay parameter of 4.5e-2. The training process is configured with a batch size of 64 across a total of 50 epochs.

E   Additional Experimental Results

We exhibit more experimental results to verify the effectiveness of our Di^{2}Pose.

E.1   Quantitative Results

As shown in Table 1, we benchmark Di^{2}Pose against SOTA 3D HPE methods on the Human3.6M under PA-MPJPE protocol. Our Di^{2}Pose achieves 39.0mm in average PA-MPJPE, surpassing the

4

Table 1: Results on Human3.6M in millimeters under PA-MPJPE. The best results are in bold, and the second-best ones are underlined.

Methods	Dir	Disc	Eat	Greet	Phone	Photo	Pose	Pur	Sit	SitD	Smoke	Wait	WalkD	Walk	WalkT	Avg
Martinez et al. [5] *ICCV'17*	39.5	43.2	46.4	47.0	51.0	56.0	41.4	40.6	56.5	69.4	49.2	45.0	49.5	38.0	43.1	47.7
Pavlakos et al. [7] *CVPR'17*	34.7	39.8	41.8	**38.6**	42.5	47.5	38.0	36.6	50.7	56.8	42.6	39.6	43.9	32.1	36.5	41.8
Liu et al. [3] *ECCV'18*	35.9	40.0	38.0	41.5	42.5	51.4	37.8	36.0	48.6	56.6	41.8	38.3	<u>42.7</u>	31.7	36.2	41.2
Zhang et al. [8] *TPAMI'23*	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	<u>39.1</u>
Choi et al. [1] *IROS'23*	36.7	41.1	37.6	42.2	40.5	44.1	37.8	36.3	47.0	60.5	39.8	38.9	42.7	33.7	35.1	40.9
Gong et al. [2] *CVPR'23*	**33.9**	**38.2**	<u>36.0</u>	<u>39.2</u>	<u>40.2</u>	**46.5**	<u>35.8</u>	<u>34.8</u>	48.0	<u>52.5</u>	41.2	<u>36.5</u>	**40.9**	<u>30.3</u>	<u>33.8</u>	39.2
Di²Pose (Ours)	<u>34.5</u>	<u>38.4</u>	**35.1**	40.8	**39.8**	<u>47.0</u>	34.9	34.7	<u>47.1</u>	52.3	40.4	36.1	42.9	30.0	33.4	**39.0**

performance of the compared SOTA 3D HPE methods, which indicates that Di²Pose is able to enhance monocular 3D HPE in indoor scenes.

E.2 Qualitative Results

In this part, we present additional qualitative results on the Human3.6M and 3DPW datasets. As illustrated in Figure 1, our Di²Pose model demonstrates the ability to accurately recover 3D human poses in both indoor and in-the-wild scenarios. Particularly noteworthy is its performance under various occlusion conditions, including self-occlusion and object occlusion. Even in these challenging situations, Di²Pose consistently produces reasonable 3D pose estimations, highlighting its robustness to occlusions.

F   Broader Impacts

This research focuses on estimating physically valid 3D human poses from monocular frames, especially under occlusion scenes. Such a method can be positively used for sports analysis, surveillance, healthcare, autonomous driving, etc. where clear, unobstructed views of the subject may not always be available. It can also lead to malicious use cases, such as illegal surveillance and video synthesis. Thus, it is essential to deploy these algorithms with care and make sure that the extracted human poses are with consent and not misused. Moreover, the diffusion-based model has a longer runtime compared to other CNN or GCN-based methods, causing more computational resources and energy consumption.

References

[1] J. Choi, D. Shim, and H. J. Kim. Diffupose: Monocular 3d human pose estimation via denoising diffusion probabilistic model. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3773–3780. IEEE, 2023.

[2] J. Gong, L. G. Foo, Z. Fan, Q. Ke, H. Rahmani, and J. Liu. Diffpose: Toward more reliable 3d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13041–13051, 2023.

[3] K. Liu, R. Ding, Z. Zou, L. Wang, and W. Tang. A comprehensive study of weight sharing in graph networks for 3d human pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 318–334. Springer, 2020.

[4] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[5] J. Martinez, R. Hossain, J. Romero, and J. J. Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 2640–2649, 2017.

[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[7] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7025–7034, 2017.

5
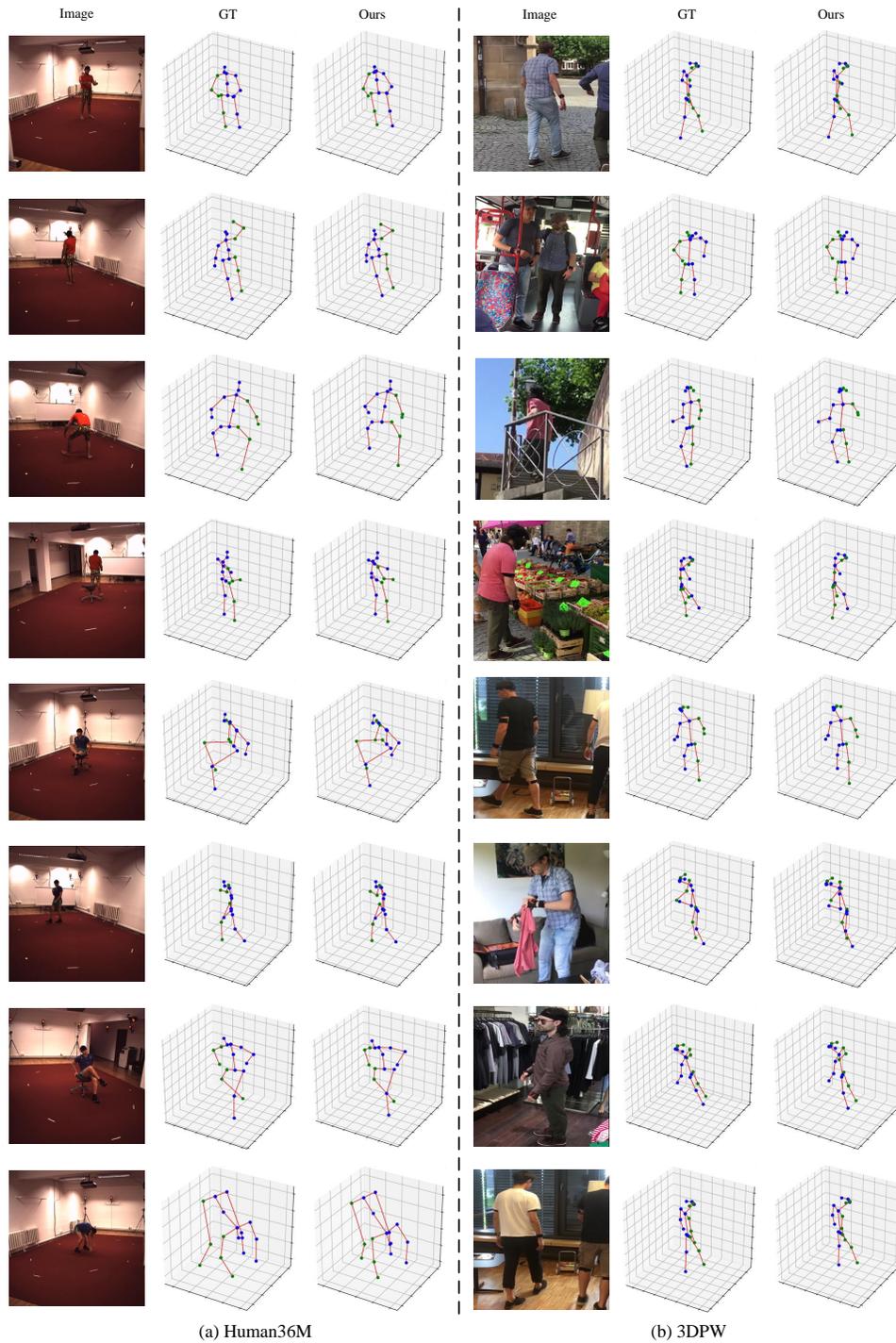
Figure 1: Qualitative results on two datasets. Joints on the right side are marked in green, while other joints are highlighted in blue.

[8] J. Zhang, K. Gong, X. Wang, and J. Feng. Learning to augment poses for 3d human pose estimation in images and videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

6