Distribution-Aligned Diffusion for Human Mesh Recovery (Supplementary)

Lin Geng Foo[1] Jia Gong[1] Hossein Rahmani[2] Jun Liu[1†]
[1]Singapore University of Technology and Design [2]Lancaster University
{lingeng_foo,jia_gong}@mymail.sutd.edu.sg, h.rahmani@lancaster.ac.uk, jun_liu@sutd.edu.sg

1. More Experiments

Here, we provide more experiments and analyses. Results are reported for 3DPW.

Visualization of the distribution denoising. In Fig. 1, we visualize the step-by-step denoising of the distributions of our method with DAT and without DAT. Using DAT, we can quickly reach the input-specific manifold, where the distribution resembles the target distribution, which shows our method can converge fast with DAT.

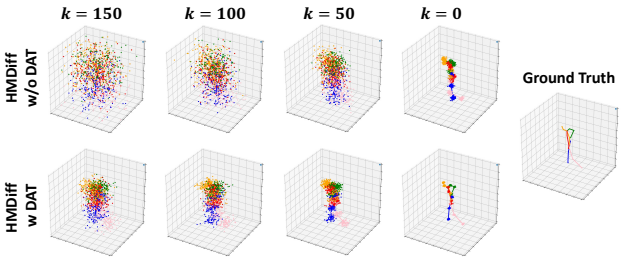

Figure 1. Visualization for DAT. For simplicity, we only show the distribution of a part of the mesh vertices.

Evaluation on occlusion benchmark. To further evaluate the effectiveness of our method in reducing uncertainty, we conduct additional experiments following [14, 6]. Specifically, we evaluate our method on 3DPW-PC [6], which is a subset of the 3DPW dataset and contains many occluded samples with high levels of uncertainty. As shown in Tab. 1, our method significantly outperforms existing methods and achieves state-of-the-art results, which shows the effectiveness of our method at handling uncertainty.

Table 1. Comparison on occlusion setting.

Method	MPVE	MPJPE	PA-MPJPE
[14]	152.8	119.7	79.7
[6]	149.6	117.5	77.1
Ours	**143.1**	**114.2**	**73.5**

Impact of the total number of diffusion steps K and number of samples N. To further investigate our HMDiff, we also conduct additional ablation experiments that vary the total number of diffusion steps K and number of samples N. From Fig. 2, we can observe that the quality of the reconstructed human mesh tends to improve when K is increased, and shows only minor improvements when K is above 200. Besides, we also find that the output mesh quality improves when we increase the number of samples N till $N = 25$, and stays roughly consistent thereafter. Therefore, we set $K = 200$ and $N = 25$.

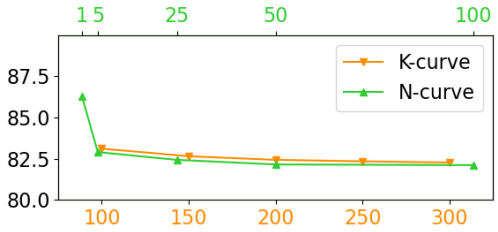

Figure 2. Evaluation of hyperparameters K and N.

More visualization results. In Fig. 2 of the main paper, we displayed some qualitative results of our method. Here, we visualize more examples in Fig. 3. These examples show that our method effectively recovers the human mesh even in scenarios with high uncertainty, such as occlusions and background noise.

Impact of acceleration technique. In our work, we accelerate the reverse diffusion process via the acceleration technique from DDIM [13], which uniformly skips diffusion steps. For more implementation details, refer to Sec. 3.2 of Supplementary. Our experiment results are presented in Tab. 2. As shown in Tab. 2, when we adopt the acceleration technique, the efficiency improves (i.e., FPS increases significantly), while the performance is similar.

Table 2. Evaluation of acceleration technique.

Method	MPVE	MPJPE	PA-MPJPE	FPS
Ours w/o acc	82.1	72.3	44.1	5
Ours w/ acc	82.4	72.7	44.5	18

† Corresponding author

Figure 3. More visualization of human mesh outputs using our method.

Impact of DAT threshold r. In Fig. 4, we evaluate the impact of different settings of the DAT threshold r. By decreasing r starting from $r = 1$ (which is equivalent to not using DAT), we observe that performance steadily improves to a peak at around $r = 0.05$, and then slightly drops. This is because, when r is high, decreasing r will lead to the distribution H_k getting aligned closer to U and H_0 (before DAT is deactivated), thus it will benefit performance. However, when r is already small (e.g., $r = 0.05$), decreasing r further will cause H_k to align towards U even at the later steps (when H_k is already relatively accurate and certain). Since the prior distribution U contains uncertainty and noise, this alignment between H_k and U at the later steps will not benefit the performance. Therefore, we fix the threshold r at 0.05.

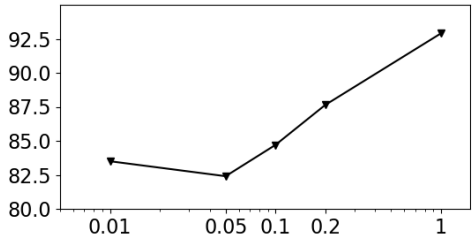

Figure 4. Evaluation of the threshold r.

2. Discussion on Theoretical Aspects

In this section, we discuss some of the theoretical aspects related to our work.

Rather than directly minimizing the gap between h_k and U by using the gradient $\nabla_{h_k} \log p(U|h_k)$, we instead compute the Distribution Alignment Gradient $\nabla_{h_k} \log p(U|\hat{h}_0(h_k))$ by using $\hat{h}_0$ as an intermediate step (as mentioned in Sec. 4.2 of the main paper). Specifically, we draw N samples from U (i.e., $u_i \sim U$) and take an update to minimize the sum of L_2 norms between $\{u_i\}_{i=1}^N$ and $f(\hat{h}_0(h_k))$. In other words, we represent $\nabla_{h_k} \log p(U|\hat{h}_0(h_k))$ with $\sum_{i=1}^N \nabla_{h_k} \log p(u_i|\hat{h}_0(h_k))$. We remark that the theoretical bounds of using $p(u|\hat{h}_0(h_k))$ to approximate $p(u|h_k)$ have been well-explored in [4].

Furthermore, following up upon the thermodynamics analogy used in our paper, more theoretical aspects regarding the thermodynamics analogy can be found in [12]. For instance, upper and lower bounds on the entropy of each reverse diffusion step have been derived in [12].

3. More Details

3.1. More network architecture details

In Sec. 4.3 of the main paper, we presented the main aspects of our network architecture. Here, we describe our network architecture in more detail.

Image feature f_I. Firstly, we go into more details regarding how we encode positional information for the image feature f_I, which is important for the transformer-based diffusion model g. Recall that we first extract a context feature $f_c \in \mathbb{R}^{2048\times7\times7}$ from our CNN backbone ϕ_I. Then, we perform average pooling on f_c and flatten it, to obtain a feature of shape 128×49. To encode positional information, we generate a position encoding map $E_s \in \mathbb{R}^{128\times49}$ – which is generated by the sinusoidal function – to add to the feature. Specifically, for the position encoding map E_s, at each index $i \in [1, ..., 49]$, $E_s[i]$ is a vector of length 128. At each even $(2j)$ index of $E_s[i]$, we set the element $E_s[i, 2j]$ to $sin(i/49^{2j/128})$, while at each odd $(2j+1)$ index, we set the element $E_s[i, 2j+1]$ to $cos(i/49^{2j/128})$. After the position encoding map E_s is added, we obtain the image feature $f_I \in \mathbb{R}^{128\times49}$, which contains rich semantic features (and positional information) and is fed to the diffusion model g at every step.

Pose estimator head ϕ_P. Our 3D pose estimator head ϕ_P (along with our CNN backbone ϕ_I) are obtained off-the-shelf from [15], where they have been pre-trained on Human3.6M [5], UP-3D [7], MuCo-3DHP [11], COCO [10] and MPII [1]. Overall. we utilize ϕ_P to generate an xy heatmap $E_{x,y} \in \mathbb{R}^{J\times56\times56}$ and a depth heatmap $E_z \in \mathbb{R}^{J\times56\times56}$, where J is the number of joints. We follow previous works [2, 9, 8] to set $J = 14$. The pose estimator head ϕ_P is a lightweight module consisting of three de-convolutional layers, in which the input feature shape in each layer is $7\times7\times1024$, $14\times14\times512$, and $28\times28\times256$ respectively.

Next, we present how to obtain a 3D human pose distribution U from ϕ_P to guide the reverse diffusion process. We note that ϕ_P allows us to generate an xy heatmap $E_{x,y} \in \mathbb{R}^{J\times56\times56}$ and a depth heatmap $E_z \in \mathbb{R}^{J\times56\times56}$, where J is the number of joints. Then, we normalize the obtained heatmaps. These normalized heatmaps can naturally be regarded as a probability map that characterizes the distribution of predicted 3D human pose. In implementation, to efficiently calculate the distribution gap between U and the intermediate distribution H_k, we sample 25 3D human poses based on these heatmaps to approximate U.

Diffusion step embedding E_d^k. Moreover, to assist the diffusion model to learn the reverse diffusion process effectively, we also build a diffusion step embedding $E_d^k \in \mathbb{R}^{61}$ for each k-th diffusion step. Specifically, at each even $(2j)$ index of E_d^k, we set the element $E_d^k[2j]$ to $sin(k/200^{2j/61})$, while at each odd $(2j+1)$ index, we set the element $E_d^k[2j+1]$ to $cos(k/200^{2j/61})$.

Diffusion model g. Overall, as described in Sec. 4.3 of the main paper, our diffusion network g is transformer-based and consists of a single vertex self-attention layer, a single vertex-image cross-attention layer, and a linear layer. V vertex tokens $\{x_1, x_2, ..., x_V\}$ are input to the network, where each token $x_v \in \mathbb{R}^{128}$ represents the v^{th} vertex of the sample h_k at step k. When the V vertex tokens $\{x_1, x_2, ..., x_V\}$ are input to the network, the vertex self-attention layer performs the self-attention operation with an adjacency matrix to generate the intermediate tokens $\{b_1, b_2, ..., b_V\}$. Then, the vertex-image cross-attention layer performs the cross-attention operation between $\{b_1, b_2, ..., b_V\}$ and f_I. Specifically, $f_I \in \mathbb{R}^{128\times49}$ is treated as 49 tokens of length 128, and this cross-attention mechanism takes $\{b_1, b_2, ..., b_V\}$ as the query components and f_I tokens as the key and value components. Next, we take the output of the cross-attention layer (of shape $V \times 128$), and feed it into a linear layer to predict $h_{k-1} \in \mathbb{R}^{V\times3}$. Then, we modify h_{k-1} via DAT (if it is activated).

MLP. Finally, after the K diffusion steps, we obtain the high-quality mesh distribution H_0, which is represented by N samples (h_0). We take the mean of H_0 by averaging the N samples, and feed it into a 3-layer MLP to obtain the final prediction h_m. More specifically, the MLP takes in a coarse human mesh sample $\mathbb{R}^{431\times3}$ and outputs a fine-grained one $\mathbb{R}^{1723\times3}$.

3.2. More implementation details

In the forward diffusion process, we set the number of total diffusion steps K at 200 and generate the decreasing sequence $\alpha_{1:K}$ via the formula:

$$\alpha_k = \prod_{i=1}^{k}(1-\beta_i) \tag{1}$$

where $\beta_{1:K}$ is a sequence from $1e-4$ to $2e-2$, which is interpolated by the linear function. Moreover, we utilize the acceleration technique DDIM [13] to speed up our diffusion inference procedure. Specifically, we accelerate our diffusion process by tuning the value of σ_k in Eq. 6 in the main paper to skip a certain number of reverse diffusion steps. Following [13], for each k-th step, we generate the acceleration metric σ_k via the formula below:

$$\sigma_k = \eta \cdot \sqrt{(1-\frac{\alpha_k}{\alpha_{k+n_i+1}}) \cdot \frac{(1-\alpha_{k+n_i+1})}{1-\alpha_k}}, \tag{2}$$

where n_i is the number of skipped steps between the current and next diffusion step, and η is the hyperparameter that controls the variance of σ_k. In our implementation, we follow [13] to set η to 0.8 and n_i to 4.

Our method is implemented using PyTorch, and can be trained on a powerful workstation with four NVIDIA RTX A5000 GPU cards within 48 hours.

In our forward and reverse process, we add noise to the mesh vertex coordinates and denoise the mesh vertex coordinates respectively. In both of these processes, the topology of the mesh vertices stays fixed. In other words, to link the mesh vertices together to produce the mesh surfaces, we follow previous works [2, 9] to define an adjacency matrix between the vertices, and this adjacency matrix is kept fixed throughout the diffusion process.

3.3. More training details

Learning Mesh Geometry. To reconstruct an accurate and natural human mesh, we also optimize our diffusion model via geometric constraints of human mesh. As mentioned in Sec. 4.4 of our main paper, following previous work [3, 8, 9], we optimize our model by incorporating four kinds of losses to learn the mesh geometry: 3D Vertex Regression Loss $\mathcal{L}_v$, 3D Joint Regression Loss $\mathcal{L}_j$, Surface Normal Loss $\mathcal{L}_n$, and Surface Edge Loss $\mathcal{L}_e$.

Specifically, we obtain the estimate of diffusion target $\hat{h}_0$ at each diffusion step and then utilize these losses to optimize the diffusion model. Below we introduce more details regarding each loss.

3D Vertex Regression Loss $\mathcal{L}_v$ is used to optimize our model to learn how to regress 3D mesh vertices. It is computed by:

$$\mathcal{L}_v = \frac{1}{V}\|\hat{h}_0 - h_0\|_1, \tag{3}$$

where V is the number of vertices and $h_0 \in \mathbb{R}^{V\times 3}$ denotes the ground-truth 3D vertex coordinates.

We can also regress 3D joints $\hat{h}_0^j$ from the estimated mesh $\hat{h}_0$ via a linear mesh-to-pose function f. Thus, we can also use a 3D Vertex Regression Loss $\mathcal{L}_j$ to train our model for the regression of 3D body joints. Specifically, this loss ($\mathcal{L}_j$) applies an L_1 loss, and is computed by:

$$\mathcal{L}_j = \frac{1}{J}\|\hat{h}_0^j - h_0^j\|_1, \tag{4}$$

where J is the number of joints and h_0^j denotes the ground-truth 3D joint coordinates.

Moreover, we optimize our model to reconstruct a consistent mesh surface via a Surface Normal Loss $\mathcal{L}_n$, which is defined as follows:

$$\mathcal{L}_n = \sum_t \sum_{\{i,j\}\subset t} \left| \left\langle \frac{\mathbf{v}_i - \mathbf{v}_j}{\|\mathbf{v}_i - \mathbf{v}_j\|_2}, n_t^* \right\rangle \right|, \tag{5}$$

where t and n_t^* denote a triangle face in the human mesh and the ground-truth unit normal vector of t respectively; $\langle\cdot,\cdot\rangle$ denotes a dot product, and $\mathbf{v}_i$ denotes the i-th vertex in t.

Finally, we utilize a Surface Edge Loss $\mathcal{L}_e$ to smooth the surfaces with dense vertices, such as face, hand and feet, The loss $\mathcal{L}_e$ is formulated as:

$$\mathcal{L}_e = \sum_t \sum_{\{i,j\}\subset t} |\|\mathbf{v}_i - \mathbf{v}_j\|_2 - \|\mathbf{v}_i^* - \mathbf{v}_j^*\|_2|, \tag{6}$$

where t and the asterisk (*) denote a triangle face in the human mesh and the ground-truth respectively, and $\mathbf{v}_i$ denotes the i-th vertex in t.

In this paper, we follow previous work [3] to set the loss coefficient values as: $\lambda_v = 0.1, \lambda_j = 1, \lambda_n = 0.0001$ and $\lambda_e = 0.005$.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693, 2014. 3

[2] Junhyeong Cho, Kim Youwang, and Tae-Hyun Oh. Cross-attention of disentangled modalities for 3d human mesh recovery with transformers. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*, pages 342–359. Springer, 2022. 3, 4

[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. In *European Conference on Computer Vision*, pages 769–787. Springer, 2020. 4

[4] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023. 2

[5] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013. 3

[6] Rawal Khirodkar, Shashank Tripathi, and Kris Kitani. Occluded human mesh recovery. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1715–1725, 2022. 1

[7] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J Black, and Peter V Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6050–6059, 2017. 3

[8] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1954–1963, 2021. 3, 4

[9] Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12939–12948, 2021. 3, 4

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 3

[11] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. Single-shot multi-person 3d pose estimation from monocular rgb. In *2018 International Conference on 3D Vision (3DV)*, pages 120–130. IEEE, 2018. 3

[12] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 2

[13] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 1, 3

[14] Yu Sun, Qian Bao, Wu Liu, Yili Fu, Michael J Black, and Tao Mei. Monocular, one-stage, regression of multiple 3d people. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11179–11188, 2021. 1

[15] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020. 3