Supplementary Materials for DANBO: Disentangled Articulated Neural Body Representations via Graph Neural Networks

In this document, we provide additional qualitative comparisons on MonoPerfCap [12] held-out sets (Section 1), highlight the improved shape reconstruction by extracting surfaces from the density field (Section 2), and show additional quantitative comparison to mesh-based and point-cloud based approaches (Section 3). We then provide the implementation and additional dataset details (Section 4), describe skeleton representation (Section 5), explain how we initialize our per-bone volumes (Section 6), and give details of our GNN design (Section 7). Finally, we give a summary on the resource consumption of DANBO compared to other baselines (Section 8). Our supplemental video in the project website shows animated results. **Real faces in all figures are blurred for anonymity.**

1 MonoPerfCap Held-out Sequences

We include qualitative comparisons on unseen pose synthesis in Figure 1, using the estimated poses from [8] as driving signals. Overall, DANBO renders consistent faces under different poses, and preserves the limbs better than A-NeRF. DANBO also synthesizes plausible wrinkles and details as visible at the clothes and jeans. Note that the driving motion (skeleton overlay) is not perfect as it is estimated with an off-the-shelf estimator.

2 Additional Geometry Comparisons

In Figure 2, we show additional results on the learned body geometry of DANBO and A-NeRF on unseen poses of the Human3.6M [3] dataset. As discussed in the main paper, DANBO reconstructs smoother and less noisy surfaces, and preserves the body part better compared to A-NeRF. We conclude that our coarse volume design acts as an regularizer to avoid learning jittery surface.

3 Additional Quantitative Comparisons on Human3.6M

We compared DANBO to NHR [11] that extracts 3D point cloud features for rendering human body and NeuralTexture [9] that synthesizes texture for 3D meshes.

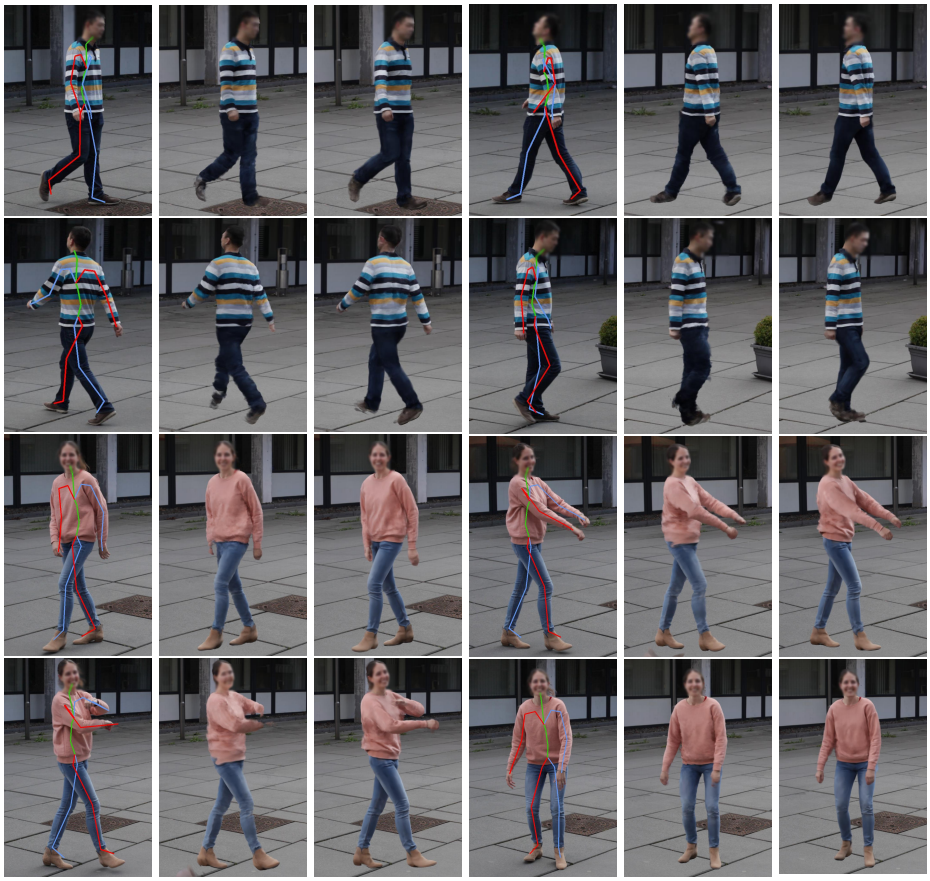

Fig. 1: **Motion retargeting on the MonoPerfCap hold-out test sets.** We overlay the reference images with the estimated poses. DANBO generates more detailed facial features and more consistent body contours, with plausible wrinkles on the jeans and clothes. **Real faces are blurred for anonymity.**

In Table 1, we report the novel view synthesis results on the testing sets. DANBO shows superior results to the two baselines that require more sophisticated 3D prior, while being self-supervised and surface-free. We show the results on unseen poses in Table 2. Similarly, DANBO outperforms the two baselines, showing better generalizability over the prior mesh-based and point clouds-based approaches.

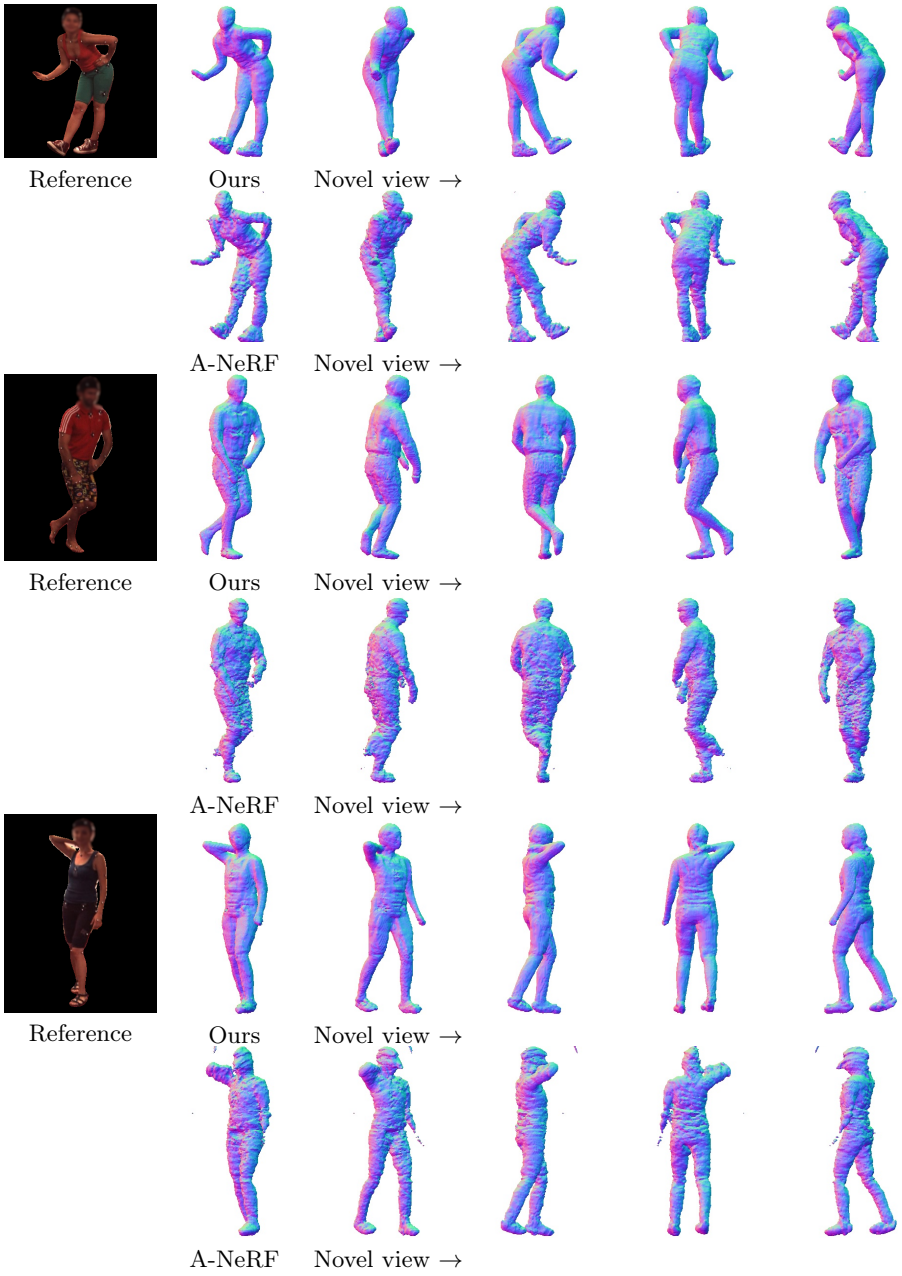

Fig. 2: **DANBO better preserves body geometry, showing a less noisy surface than A-NeRF.** We extract the isosurface using Marching cubes [4] with voxel resolution 256.

Table 1: **Novel-view synthesis comparisons on Human3.6M [3]**. DANBO achieves higher novel view synthesis quality compared to point clouds-based method NHR and mesh-based approach NeuralTexture.

	NHR[11]				NeuralTexture [9]				**DANBO (Ours)**			
	PSNR ↑	SSIM ↑	KID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	KID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	KID ↓	LPIPS ↓
S1	21.79	0.892	0.094	0.155	22.04	0.893	0.054	0.150	**23.95**	**0.916**	**0.033**	**0.148**
S5	21.37	0.892	0.050	0.145	20.91	0.887	0.036	0.144	**24.86**	**0.924**	**0.029**	**0.142**
S6	21.05	0.854	0.050	0.145	21.10	0.855	0.039	0.145	**24.54**	**0.903**	**0.035**	**0.143**
S7	21.11	0.890	0.066	0.137	21.59	0.891	0.042	0.133	**24.45**	**0.920**	**0.028**	**0.131**
S8	19.80	0.889	0.052	0.177	17.69	0.871	**0.026**	0.174	**23.36**	**0.917**	0.068	**0.173**
S9	23.77	0.898	0.089	0.144	23.80	0.899	0.085	0.139	**26.15**	**0.925**	**0.040**	**0.137**
S11	22.65	0.891	0.108	0.160	22.73	0.892	0.076	0.155	**25.58**	**0.917**	**0.060**	**0.153**
Avg	21.65	0.887	0.073	0.152	21.41	0.884	0.051	0.149	**24.70**	**0.917**	**0.042**	**0.146**

Table 2: **Novel-pose synthesis comparisons on Human3.6M [3]**. DANBO outperforms the NHR and NeuralTexture on novel pose synthesis despite not relying on 3D supervisions for training.

	NHR[11]				NeuralTexture [9]				**DANBO (Ours)**			
	PSNR ↑	SSIM ↑	KID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	KID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	KID ↓	LPIPS ↓
S1	21.24	0.877	0.171	0.142	21.08	0.874	0.128	0.137	**23.03**	**0.895**	**0.081**	**0.135**
S5	21.55	0.883	0.060	0.151	21.06	0.879	**0.048**	0.150	**23.66**	**0.903**	0.049	**0.147**
S6	21.14	0.876	0.075	0.162	21.20	0.875	0.090	0.160	**24.57**	**0.906**	**0.052**	**0.158**
S7	20.52	0.878	0.068	0.143	21.15	0.878	0.048	0.137	**23.08**	**0.897**	**0.036**	**0.136**
S8	19.55	0.876	0.064	0.171	17.67	0.862	**0.049**	0.167	**22.60**	**0.904**	0.092	**0.167**
S9	23.02	0.883	0.086	0.143	23.10	0.883	0.097	0.138	**24.79**	**0.904**	**0.042**	**0.136**
S11	22.94	0.890	0.080	0.149	22.72	0.889	0.061	0.147	**24.57**	**0.901**	**0.040**	**0.144**
Avg	21.42	0.880	0.086	0.152	21.14	0.877	0.074	0.148	**23.76**	**0.902**	**0.056**	**0.146**

4 Implementation and Dataset Details

Implementation. We train our model for 200k iterations, with the same learning rate decay schedule as in [6]. We use a hyperparameter setting similar to that in A-NeRF [8], with several exceptions. For each training batch, we sample 3072 rays from 16 images with 96 uniform samples and 48 importance samples along each ray. This results in the same number of network evaluations as A-NeRF. Additionally, unlike the original NeRF, we learn a single neural field instead of two separate coarse and fine networks. We use $M = 16$ and $H = 5$ for the factorized volumes in all experiments unless stated otherwise. Training takes around 20 hours on 2 NVidia V100 GPUs. Following prior work, we also optimize a 128-dim latent code for each frame to model the appearance changes that cannot be explained by the skeleton pose alone [5,7,8]. The total GPU memory consumption during training is around 11.0G.

Dataset. Below, we provide additional descriptions on Mixamo [1] and Surreal+CMU-Mocap [10,2] datasets:

– **Mixamo** [1] is a synthetic dataset that includes challenging dancing motions. We use the sequences processed by [8] for motion retargeting.

– **Surreal+CMU-Mocap** [10] is a synthetic dataset animated using real-life motion capture data CMU-Mocap [2]. It consists of diverse motion sequences such as gymnastic poses and ballet dances, making it a suitable dataset for testing out-of-distribution novel pose synthesis.

5 Skeleton Representation

Our skeleton representation consists of a rest pose $\mathbf{J} = [\mathbf{j}_1, \mathbf{j}_2, \cdots, \mathbf{j}_{24}]$ of 3D joint locations with fixed connectivity and bone lengths, and the skeleton pose $\theta = [\omega_1, \omega_2, \cdots, \omega_{24}]$ that models the per-frame motion via the rotation ω_i relative to their parents. The root is at $i = 1$ with ω_i encodes the global rotation. The local-to-world transformation that maps a 3D point from the local coordinate of i to the world space is computed via forward kinematic with homogeneous coordinates

$$\mathbf{G}(\omega_i) = \prod_{l \in A(i)} \begin{bmatrix} R(\omega_l) & \mathbf{j}_{l,l-1} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbf{R}^{4\times4}, \tag{1}$$

where R is the 3×3 rotation matrix converted from the 6D rotation [13], $A(i)$ is the ordered set of the joint ancestors of i, and $\mathbf{j}_{l,l-1}$ is the translation between l and its child joint $l-1$. The world-to-local transformation is then the inverse of Equation (1),

$$T(\omega_i) = \mathbf{G}(\omega_i)^{-1} \in \mathbf{R}^{4\times4}. \tag{2}$$

In practice, since we want to center the per-bone volume in between joint i and its child j for the best volume coverage, we rewrite Equation (2)

$$T(\omega_i) = \hat{T}(\omega_i)\mathbf{G}(\omega_i)^{-1} \in \mathbf{R}^{4\times4}, \text{ where } \hat{T}(\omega_i) = \begin{bmatrix} I & 0.5\mathbf{j}_{j,i} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3}$$

with $I \in \mathbf{R}^{3\times3}$ be an identity matrix We set $\mathbf{j}_{j,i} = \mathbf{0}$ when i has multiple or no child. Intuitively, $\hat{T}$ shifts the coordinates from joint i to the middle of the bone defined by i and j. Note that our transformation $T(\omega_i)$ also aligns the vector from joint i to its child j with z-axis.

6 Per-bone Volume initialization

We use a set of simple heuristic for initializing the volume scale (s_i^x, s_i^y, s_i^z). We first split the skeleton into 4 sections, namely torso, leg, arm, and head. We then leverage the distances between three symmetric joints in the rest pose, D_{shoulder}, D_{collar}, and D_{knee}, as references to initialize volume width s_i^x and height s_i^y. We initialize volume length s_i^z using the bone length $||\mathbf{j}_{i,j}||$ between joint i and its child joint j. For joints with multiple children, we use the mean bone lengths to all children instead. And for leaf nodes like the head and hands, we use the maximum bone length $||\mathbf{j}_{\max}||$ of the whole skeleton for initializing s_i^z. We illustrate the body sections and reference distances in Figure 3, and report the

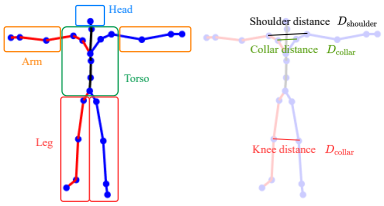

Fig. 3: We split the skeleton into 4 different sections: head (blue), arm (orange), torso (green) and leg (red). We initialize the volumes from each section with values derived from shoulder width, collar width, and knee width.

Table 3: We initialize the volume width, height and length using D_{shoulder}, D_{collar}, D_{knee} and bone length $||\mathbf{j}_{i,j}||$ as heuristic. Note s_i^z is aligned with the bone direction of i.

	s^x	s^y	s^z
Head	$1/(0.6D_{\text{shoulder}})$		$1/(1.1\|\|\mathbf{j}_{\text{max}}\|\|)$
Torso	$1/(0.7D_{\text{shoulder}})$		$1/\|\|\mathbf{j}_{i,j}\|\|$
Arm	$1/(0.6D_{\text{collar}})$		$1/\|\|\mathbf{j}_{i,j}\|\|$ or $1/\|\|\mathbf{j}_{\text{max}}\|\|$ for leaf
Leg	$1/(0.5D_{\text{knee}})$		$1/\|\|\mathbf{j}_{i,j}\|\|$ or $1/\|\|\mathbf{j}_{\text{max}}\|\|$ for leaf

detail initialization configuration in Table 3. Our heuristic initializes per-bone volumes to be large enough to encompass the whole human body, and at the same time be as small as possible for subsequent minimization to reduce empty space within the volumes. Recall that this is only an initialization. The exact scale is optimized alongside the neural radiance field.

7 GNN and Per-node MLP

GNN. We treat the human skeleton as a graph, where each node representing one bone. We initialize the node embedding $\mathbf{n}_i^{(0)}$ of i as the associated bone rotation $\omega_i \in \mathbb{R}^6$ in the 6D form [13]. The graph convolutional layer $l = 1, \cdots, L$ then updates the node embedding by

$$\mathbf{m}_i^l = \mathbf{W}_i^{(l)} \mathbf{n}^{(l-1)}, \tag{4}$$

$$\mathbf{n}_i^{(l)} = \text{ReLU}\left(\left(\sum_{a \in \mathcal{A}(i) \cup \{i\}} \mathbf{A}_{(i,a)}^{(l)} \mathbf{m}_a^l\right) + \mathbf{b}^{(l)}\right), \tag{5}$$

where $\mathbf{m}_i^l$ is the *message* projected by the weight matrix $\mathbf{W}_i^{(l)}$ of bone i, and the updated node embedding $\mathbf{n}_i^{(l)}$ is computed by aggregating the messages from the 1-hop neighbors $\mathcal{A}(i)$. $\mathbf{A}^{(l)}(i,a)$ is a learnable scalar for controlling the message flow from i to a, and $\mathbf{b}^{(l)}$ is the bias term shared among all nodes. In all our experiments, we use $L = 2$ graph convolutional layers, $\mathbf{n}_i^{(l)} \in \mathbf{R}^{128}$ for $l > 0$, and initialize $\mathbf{A}_{(i,i)}^{(l)} = 1$ and $\mathbf{A}_{(i,a)}^{(l)} = \min(0.05 + \epsilon, 0.01)$ with $\epsilon \sim \mathcal{N}(0, 0.1)$. Note that the GNN can potentially achieve better performance by using the per-node bias and making $\mathbf{A}_{(i,a)}^{(l)}$ conditioned on θ, but we forgo these options for simplicity. We use per-node weight matrix $\mathbf{W}_i^{(l)}$ since the graph is heterogeneous: each node (body part) has attributes different from the others.

Per-node MLP. As mentioned above, the skeleton graph is heterogeneous. We therefore use per-node MLPs to learn weights specific for individual body parts. For the part-disentangled graph neural network G, we use 2-layer per-node MLPs after the two graph convolutional layers. The aggregation network follows the same design, but using only one graph convolutional layer.

8 Resource Consumption

We compare to A-NeRF with layer width 448 (2.3M parameters) to match ours in capacity (2.5M) for a fair comparison. Using the factorized volume has a much lower parameter count than the conventional 3D grid (4.9M). Anim-NeRF has smaller capacity (1.3M) and optimize a 128-dim latent vector for each pose to predict the deformation fields. NeuralBody has the largest model capacity (5.0M). On the Nvidia V100 we used for training, DANBO takes on average 170 ms for every gradient descent iteration and around 6 seconds for synthesizing a 512×512 image. This is faster than A-NeRF with comparable capacity (around 8 seconds) but slower than Anim-NeRF (around 2 seconds).

References

1. Adobe: Mixamo. https://www.mixamo.com/ (2020)
2. CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu
3. Ionescu, C., Carreira, J., Sminchisescu, C.: Iterated Second-Order Label Sensitive Pooling for 3D Human Pose Estimation. In: CVPR (2014)
4. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM TOG (Proc. SIGGRAPH) (1987)
5. Martin-Brualla, R., Radwan, N., Sajjadi, M.S.M., Barron, J.T., Dosovitskiy, A., Duckworth, D.: NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In: CVPR (2021)
6. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
7. Peng, S., Zhang, Y., Xu, Y., Wang, Q., Shuai, Q., Bao, H., Zhou, X.: Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In: CVPR (2021)
8. Su, S.Y., Yu, F., Zollhöfer, M., Rhodin, H.: A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. In: NeurIPS (2021)
9. Thies, J., Zollhöfer, M., Nießner, M.: Deferred neural rendering: Image synthesis using neural textures. ACM TOG (Proc. SIGGRAPH) (2019)
10. Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M.J., Laptev, I., Schmid, C.: Learning from synthetic humans. In: CVPR (2017)
11. Wu, M., Wang, Y., Hu, Q., Yu, J.: Multi-view neural human rendering. In: CVPR (2020)
12. Xu, W., Chatterjee, A., Zollhöfer, M., Rhodin, H., Mehta, D., Seidel, H.P., Theobalt, C.: Monoperfcap: Human performance capture from monocular video. TOG **37**(2), 27 (2018)
13. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: CVPR. pp. 5745–5753 (2019)