Supplementary Material: Learning Motion-Dependent Appearance for High-Fidelity Rendering of Dynamic Humans from a Single Camera

Jae Shin Yoon[†,♯] Duygu Ceylan[♯] Tuanfeng Y. Wang[♯]
Jingwan Lu[♯] Jimei Yang[♯] Zhixin Shu[♯] Hyun Soo Park[†]

[†]University of Minnesota [♯]Adobe Research

In the supplementary materials, we provide the details and evaluations of our 3D performance tracking method, network designs for our human rendering models, and more results and comparison. Please also refer to the supplementary video.

A. Model-based Monocular 3D Pose Tracking

Given a video of a moving person, we represent $\mathbf{p}$ as the posed 3D body at each frame. Specifically, we predict the parameters of the template SMPL model [8], i.e., $\mathbf{p} = SMPL(\boldsymbol{\theta}, \boldsymbol{\beta})$, where $SMPL$ is a function that takes the pose $\boldsymbol{\theta} \in \mathbb{R}^{72}$ and shape $\boldsymbol{\beta} \in \mathbb{R}^{10}$ parameters and provides the vertex locations of the 3D posed body. To this end, we learn a tracking function that regresses accurate and temporally coherent pose and camera parameters from an image sequence:

$$\boldsymbol{\theta}_t, \mathbf{C}_t = f_{\text{track}}(\mathbf{A}_t), \tag{1}$$

where f_{track} is the tracking function, $\mathbf{A}_t$ is the image at time t, and $\mathbf{C}_t \in \mathbb{R}^3$ is the camera translation relative to the body, camera rotation is encoded in $\boldsymbol{\theta}_t$. We assume the shape, $\boldsymbol{\beta}$, is constant. We use a weak-perspective camera projection model [4] where we represent the camera translation in the z axis as the scale parameter. f_{track} is learned by minimizing the following loss for each input video:

$$\mathcal{L}_{\text{track}} = \mathcal{L}_{\text{f}} + \lambda_{\text{r}}\mathcal{L}_{\text{r}} + \lambda_{\text{d}}\mathcal{L}_{\text{d}} + \lambda_{\text{t}}\mathcal{L}_{\text{t}}, \tag{2}$$

where $\mathcal{L}_{\text{f}}$, $\mathcal{L}_{\text{r}}$, $\mathcal{L}_{\text{d}}$, and $\mathcal{L}_{\text{t}}$ are the fitting, rendering, data prior, and temporal consistency losses, respectively, and λ_{r}, λ_{d}, and λ_{t} are their weights. We set $\lambda_{\text{r}} = 1, \lambda_{\text{d}} = 0.1$, and $\lambda_{\text{t}} = 0.01$ in our experiments. The overview of our optimization framework is described in Figure 1.

L_{f} and $\mathcal{L}_{\text{r}}$ utilize image-based dense UV map predictions [3] which enforce the 3D body fits to better align with the image space silhouettes of the body. Specifically, L_{f} measures the 2D distance between the projected 3D vertex

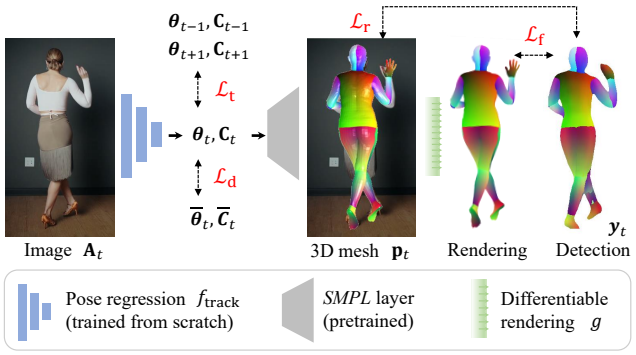

Figure 1. The overview of our model-based monocular 3D performance tracking. A regression network predicts the body ($\boldsymbol{\theta}$) and camera ($\mathbf{C}$) pose parameters from a single image. The pretrained SMPL layer [8] decodes the predicted parameters to reconstruct the posed 3D body mesh. We render out the dense IUV coordinates of the mesh using a differentiable rendering layer and train the regression network by enforcing self-consistency between densepose detection and rendered IUV map [3] ($\mathcal{L}_{\text{r}}$ and $\mathcal{L}_{\text{f}}$); and enforcing temporal smoothness ($\mathcal{L}_{\text{t}}$) and data-driven regularization ($\mathbf{L}_{\text{d}}$).

locations and corresponding 2D points in the image:

$$\mathcal{L}_{\text{f}} = \sum_{\mathbf{X} \leftrightarrow \mathbf{x} \in \mathcal{U}} \|\Pi_p \mathbf{X} - \mathbf{x}\|. \tag{3}$$

where $\mathcal{U}$ is the set of dense keypoints in the image, $\mathbf{x} \in \mathbb{R}^2$ obtained from image-based dense UV map predictions [10], $\mathbf{X}$ are the corresponding 3D vertices, and Π_p is the camera projection which is a function of $\mathbf{C}$. $\mathcal{L}_{\text{r}}$ measures the difference between the rendered and detected UV maps, $\mathbf{y}$:

$$\mathcal{L}_{\text{r}} = \|g(\mathcal{W}^{-1}\mathbf{p}^t, \mathbf{C}_t) - \mathbf{y}\|, \tag{4}$$

where $g(\cdot)$ is the differentiable rendering function that renders the UV coordinates from the 3D body model.

$\mathcal{L}_{\text{d}}$ provides the data driven prior on body and camera poses, i.e., $\mathcal{L}_{\text{d}} = \|\boldsymbol{\theta} - \overline{\boldsymbol{\theta}}\| + \|\mathbf{C} - \overline{\mathbf{C}}\|$, where $\overline{\boldsymbol{\theta}}$ and $\overline{\mathbf{C}}$ are the initial body and camera parameters predicted by a state-of-the-art method [5]. $\mathcal{L}_{\text{t}}$ enforces the temporal smoothness

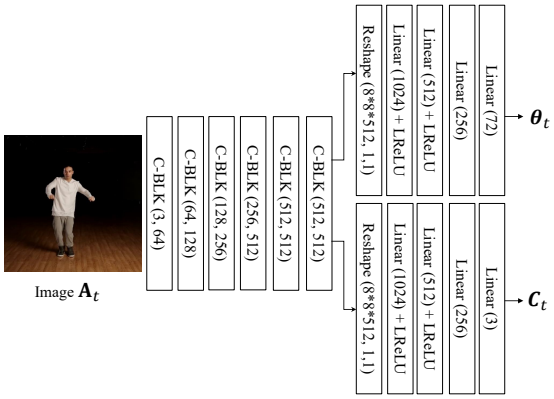

Figure 2. Network design for our 3D body and camera pose regression network (f_{track}). The details for C-BLK, D-BLK, Conv, and LReLU are described in Figure 3.

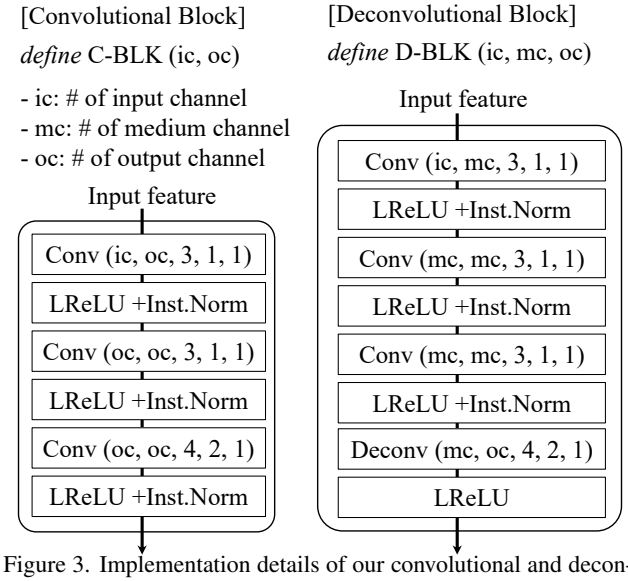

Figure 3. Implementation details of our convolutional and deconvolutional blocks. Conv and Deconv denotes convolutional and deconvolutional layers are constructed based on the parameters: number of input channels (ic), number of output channels (oc), filter size, stride, and the size of the zero padding. We set the coefficient of the LeakyReLU (LReLU) to 0.2.

over time: $\mathcal{L}_t = \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\| + \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}\| + \|\mathbf{C}_t - \mathbf{C}_{t-1}\| + \|\mathbf{C}_t - \mathbf{C}_{t+1}\|$.

We enable f_{track} using a convolutional neural network. The details of our network designs are described in Figure 2. where it predicts the 3D body $\boldsymbol{\theta}$ and camera $\mathbf{C}$ pose from an image $\mathbf{A}$.

Evaluation We validate the performance of our 3D pose tracking method by comparing with previous monocular image based (SPIN [6] and SMPLx [2]) and video based (VIBE [5]) 3D body estimation methods.

We use the AIST++ dataset [7] which provides pseudo-ground truth SMPL fits obtained from multiview images. For randomly selected four subjects, we select four viewpoints and two motion styles (600 frames per motion) resulting in 4800 testing frames per subject. Due to the differences in the camera models adopted by each method (*i.e.,* perspective or orthographic cameras), there exist a scale ambiguity between the predictions and the ground truth. Hence, we measure the per-vertex 2D projection error between the ground truth and predicted 3D body model in the image space. We provide quantitative and qualitative results in Table 1 and Figure 4, respectively. By exploiting both temporal cues and dense keypoint estimates, our method outperforms the previous work.

	Sub.1	Sub.2	Sub.3	Sub.4	Avg.
SPIN [6]	16.5±3.7	22.6±6.6	23.4±6.2	21.5±4.4	21.0±5.2
VIBE [5]	13.9±2.9	12.2±2.8	17.7±5.1	15.5±2.9	14.8±3.4
SMPLx [2]	9.0±1.6	10.2±1.7	16.2±10.2	12.1±4.4	11.9±4.5
Ours	8.3±1.1	8.7±2.0	13.7±3.5	11.3±1.9	10.5±2.1

Table 1. We show the mean and std of per-vertex projection error between the ground truth and estimated 3D bodies for images of size 512×512.

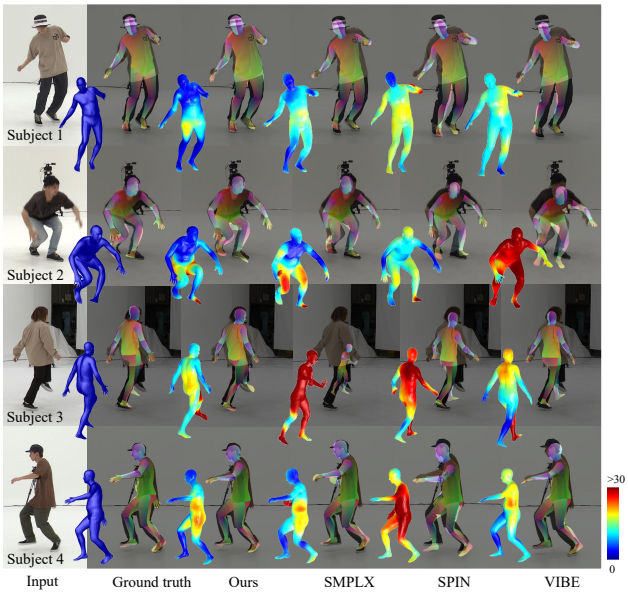

Figure 4. We show the 3D body estimates and color coded 2D projection errors of our method and baselines for images of size 512×512.

B. Comparison to 3D based Approach

In Fig. 5 and Table 2, we show qualitative and quantitative comparisons to the recent 3D based method [1] for neural avatar modeling from a single camera, which explicitly reconstruct the geometry of animatable human. This method cannot effectively produce motion-dependent texture due to the failure in modeling of 3D deformation for clothing geometry, leading to rendering results with blurry, noisy, and static appearance as shown in Fig. 5, upper row.

Figure 5. Qualitative comparison to NeuralAvatar [1]. The color map represents the per-pixel difference from real images.

	MPI	*Custom* 1	*Custom* 2
NeuralAvatar [1]	0.808 / 15.3	0.860 / 12.2	0.869 / 12.7
Ours	**0.825 / 2.82**	**0.942 / 3.12**	**0.946 / 3.81**

Table 2. Comparison to the 3D method. Two metrics represent SSIM (↑), LPIPS (↓)×100, respectively. Three datasets are used.

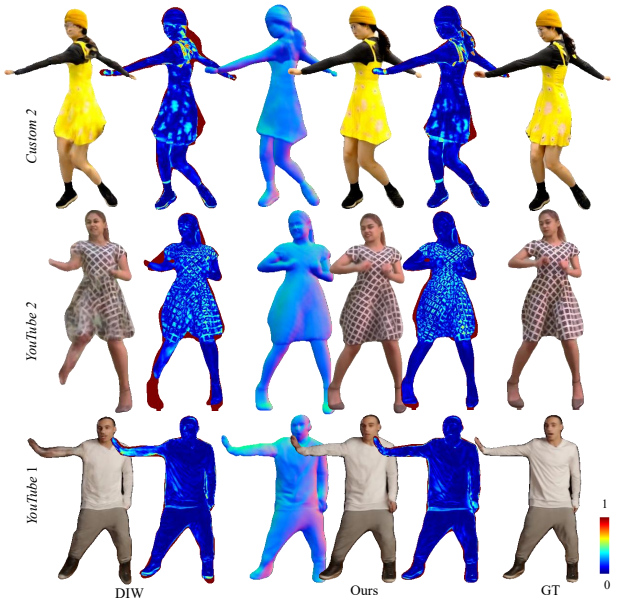

Figure 6. Results from the model that learns from a small amount of data (10 % of full training data). The color map shows the pixel-wise difference of the synthesized image with the ground truth.

C. More results

In Figure 6, we show the qualitative results of our rendering model and the one from DIW [9] which are trained with a small number of data (10% of the full training data). This result shows a similar trend as the Table 2 in the main manuscript.

D. Network Designs for Human Rendering

We learn our motion encoder E_Δ and compositional rendering decoders, E_s, E_a using convolutional neural networks. In this section, we provide the implementation details of our network designs.

3D Motion Encoder Network, E_Δ. Figure 7 describes the network details for our 3D motion encoder E_Δ. It takes as input 3D surface normal $\mathbf{N}_t$ of the current frame and velocity $\mathbf{V}_t$ for past 10 frames recorded in the UV space of the body and outputs 3D motion descriptors $\mathbf{f}_{3\mathrm{D}}^t$.

Shape Decoder Network, E_s. Figure 8 describes the network details for our shape decoder network E_s which takes as input the 3D motion descriptor $\widehat{f}_t$ rendered in the image space and the predicted shape in the previous time instance $\widehat{\mathbf{s}}_{t-1}$, and outputs the person-specific 2D shape $\widehat{\mathbf{s}}_t$ which is composed seven category label maps.

Appearance Decoder Network, E_a. Figure 9 describes the network details for our appearance decoder network E_s which takes as input the projected 3D motion descriptor $\widehat{f}_t$ rendered in the image space, predicted shape $\widehat{\mathbf{s}}_t$, and the predicted appearance and surface normal in the previous time instance $\{\widehat{\mathbf{A}}_{t-1}, \widehat{\mathbf{n}}_{t-1}\}$, and outputs the 3D surface normal $\widehat{\mathbf{n}}_t$ and appearance $\widehat{\mathbf{A}}_t$.

References

[1] Jianchuan Chen, Ying Zhang, Di Kang, Xuefei Zhe, Linchao Bao, and Huchuan Lu. Animatable neural radiance fields from monocular rgb video. *ICCV*, 2021. 2, 3

[2] Vasileios Choutas, Georgios Pavlakos, Timo Bolkart, Dimitrios Tzionas, and Michael J Black. Monocular expressive body regression through body-driven attention. In *ECCV*, 2020. 2

[3] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *CVPR*, pages 7297–7306, 2018. 1

[4] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. 1

[5] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *CVPR*, 2020. 1, 2

[6] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *ICCV*, 2019. 2

[7] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++, 2021. 2

[8] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *ACM TOG*, 2015. 1

[9] Tuanfeng Y. Wang, Duygu Ceylan, Krishna Kumar Singh, and Niloy J. Mitra. Dance in the wild: Monocular human animation with neural dynamic appearance synthesis, 2021. 3

[10] Yuanlu Xu, Song-Chun Zhu, and Tony Tung. Denserac: Joint 3d pose and shape estimation by dense render-and-compare. In *ICCV*, pages 7759–7769, 2019. 1

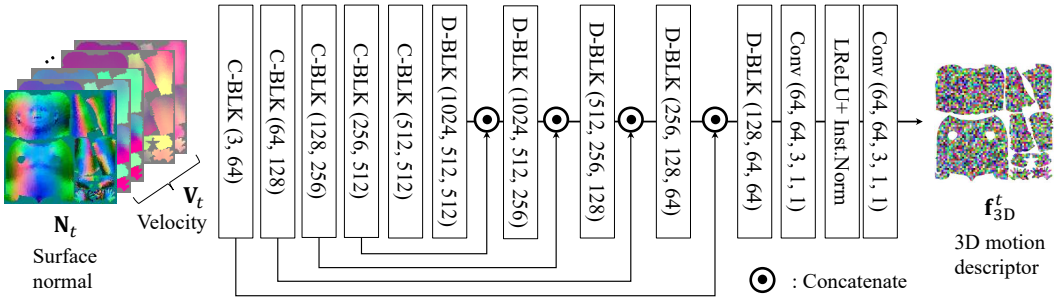

Figure 7. Network design for our 3D motion encoder ($E_{\triangle}$). The details of C-BLK, D-BLK, Conv, and LReLU are described in Figure 3.

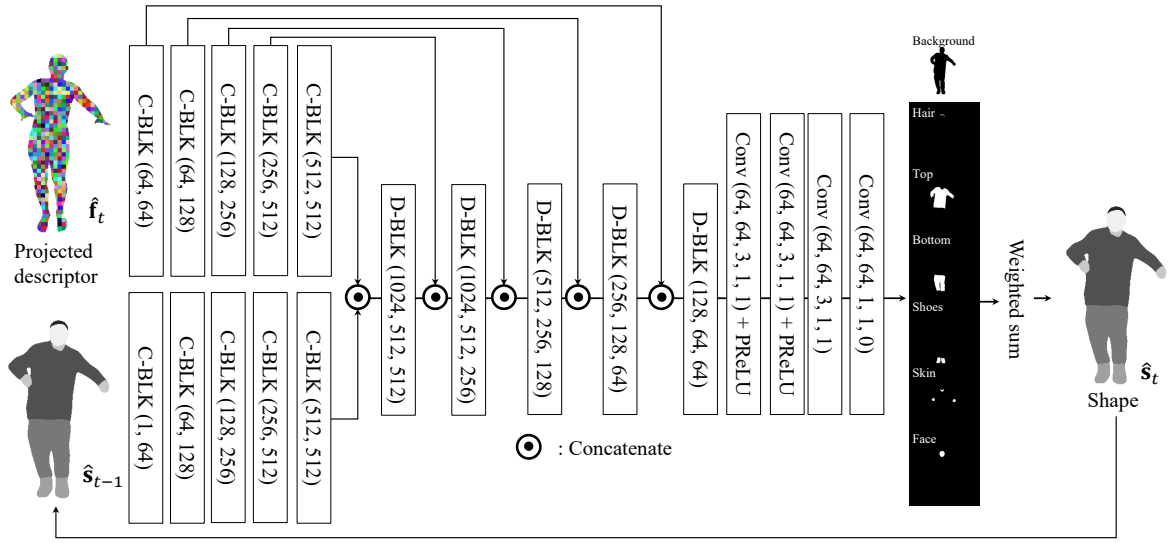

Figure 8. Network design for our shape decoder (D_s). The details of C-BLK, D-BLK, Conv, and LReLU are described in Figure 3.

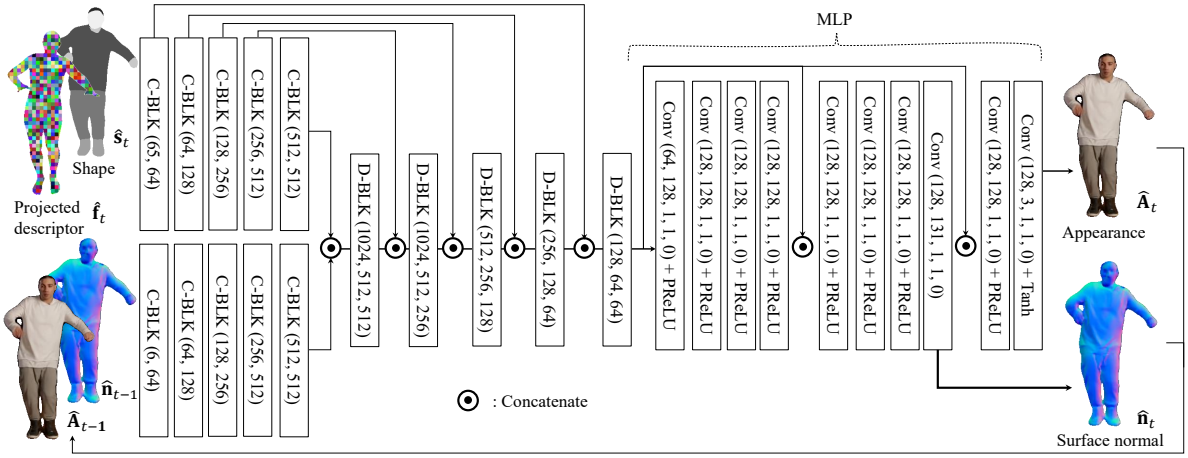

Figure 9. Network design for our appearance decoder (D_a). The details of C-BLK, D-BLK, Conv, and LReLU are described in Figure 3.