Modeling Clothing as a Separate Layer for an Animatable Human Avatar (Supplementary Document)

1 CLOTHED BODY REGISTRATION

In this section, we give a detailed description of our clothed body registration pipeline (Sec. 4 of the main paper).

1.1 Data Preprocessing

The input to our pipeline is a sequence of RGB images of the subject captured by a synchronized multi-camera system. The raw RGB images are used to create a dense 3D reconstruction of the human surface with a multi-view Patchmatch reconstruction algorithm [Galliani et al. 2015]. One example of reconstructed mesh can be seen in Fig. 1.

Then, we run body part segmentation for the input images. For the network architecture we use PointRend [Kirillov et al. 2020], which takes raw RGB images as input and outputs segmentation masks of the same size. We also run keypoint detection for body and hand joints. We use a multi-stage 2D pose estimation network similar to [Tan et al. 2020] to first detect 2D keypoints from raw RGB images, then perform multi-view triangulation to obtain 3D keypoints. Both the segmentation and keypoint detection models are trained on our in-house datasets consisting of images captured in the aforementioned multi-camera system. Additionally the 2D pose estimation model is pretrained on the MS COCO dataset [Lin et al. 2014]. An example of part segmentation and the detected keypoints can be seen in Fig. 1.

We aim to separately register body and clothing templates to the above observations, including images, scan, segmentation masks and keypoints. First, we run single-layer surface registration to fit to the scan. Then, we segment the single-layer registered meshes into different body and clothing regions based on the multi-view 2D segmentation masks. After that, we register the clothing template to the segmented surface from the single-layer registration with an explicit boundary-aware loss. Finally, we estimate the underlying body shape in the inner layer.

1.2 Single-Layer Surface Tracking.

We non-rigidly register the reconstructed meshes with a kinematic body model, similar to [Zhang et al. 2017] and [Walsman et al. 2017]. We use a kinematic body model with $N_j = 159$ joints, $N_v = 614,118$ vertices and pre-defined LBS skinning weights for all the vertices. Let $\mathcal{W}(\cdot,\cdot)$ be the LBS function that takes rest-pose vertices and joint angles as input, and outputs the target-pose vertices. First, we estimate a personalized model by computing the rest-state shape $\mathbf{V}_i \in \mathbb{R}^{N_v \times 3}$ that best fit a collection of manually selected peak poses. Then, for each frame i, we estimate a set of joint angles $\boldsymbol{\theta}_i$, such that the skinned mesh $\hat{\mathbf{V}}_i = \mathcal{W}(\mathbf{V}_i, \boldsymbol{\theta}_i)$ best matches the 3D reconstruction and detected keypoints introduced in the data preprocessing step. Finally, we introduce additional per-frame vertex offsets on top of the skinned kinematic model. We minimize the distance between the deformed surface and the 3D reconstruction with Laplacian regularization. At this point, we have obtained registered meshes representing the human surface as a whole, which provides a prior for solving the two layer registration problem.

1.3 Clothing Registration

Our clothing registration step produces aligned geometry for the outer layer, and is similar in spirit to [Pons-Moll et al. 2017]. We briefly explain the process for completeness and focus on the differences. The process uses the single-layer surface tracking results and clothing part segmentation in all camera views as input. In detail, it consists of the following steps.

Mesh segmentation. Here we aim to label each vertex in the tracked surface mesh as either 'outer' or 'inner'. Because we only model the clothing on the upper body in the outer layer, we label only the vertices in the upper body clothing region as 'outer', and all other regions, including exposed skin and pants, as 'inner'. To identify the vertices belonging to the upper body clothing, we project the mesh to each camera view and check the clothing part segmentation in the projected pixel location. The majority vote of segmentation labels among different camera views gives us the initial vertex segmentation labels. We also filter out vertices visible in less than three camera views and leave them as undetermined, which happens frequently in the region below the armpit. Similar to [Pons-Moll et al. 2017], we then use Markov Random Fields (MRF) to refine the initial vertex segmentation results. This approach allows us to fill in the undetermined region and remove noisy labelling in the initial segmentation results. The output of MRF gives us the binary inner/outer per-vertex segmentation results.

Clothing template creation. We manually select one frame of single-layer tracking results, and use the upper clothing region identified in the mesh segmentation step as our clothing template. We ask an artist to create a UV map for the template, which is used to render the clothing and to represent the clothing geometry in the UV space in the codec avatar (see Fig. 3). Each vertex in the clothing template is associated with a vertex in the whole-body model $\mathbf{V}_i$, and can be skinned using the same kinematic joints and LBS weights. We also reuse the triangulation in the whole-body model to create a topology for the clothing template. In the following steps, we will use this template to register the clothing shape for all other frames in the sequence.

Deformation initialization. For each frame in the sequence, we use the mesh segmentation results to identify the target region of upper clothing. We also initialize a clothing mesh using the template topology created in the previous step. The goal is to deform the template mesh to match the target clothing mesh. In order to provide better initialization for the deformation, we apply biharmonic deformation fields [Jacobson et al. 2010] to find a per-vertex deformation that aligns the boundary of the template mesh to the target mesh boundary while keeping the interior distortion as low as possible.

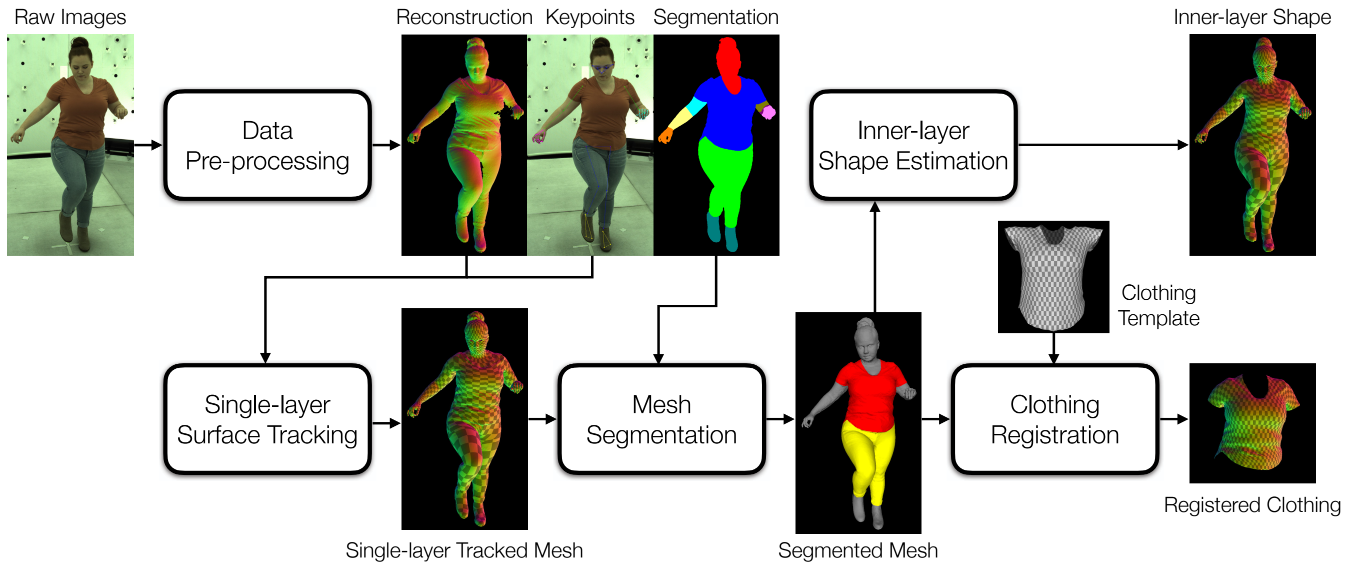

Fig. 1. Clothed body registration pipeline.

We observe that this allows the template shape to converge to a better minimum in the following iterative optimization step.

Boundary-aware non-rigid ICP. Given the deformation initialization results from the last step, we then run non-rigid Iterative Closest Points (ICP) algorithm to register the template mesh to the target clothing mesh, with a special focus on the boundary alignment. Similar to Section 5.3.1 in [Pons-Moll et al. 2017], we optimize a weighted sum of the ICP term, the Laplacian term for mesh regularization and a boundary term that penalizes the distance between the boundary vertices of the template mesh and the target mesh.

After these steps, we obtain a separate mesh registered to the upper clothing of the subject in the same template topology for each frame in the sequence, which we later use to train the clothing branch of our codec avatar.

1.4 Inner-Layer Shape Estimation

In this section, we describe our method to estimate the registered shape for the inner layer of the codec avatar. The inner layer geometry consists of two parts: invisible body region covered by the upper clothing, which we needs to estimate in this section, and other visible region of the human surface, which can be directly obtained from the single-layer whole-body tracking results in Sec. 1.2. Our method to estimate the inner-layer shape is similar to that in [Zhang et al. 2017], which estimates an underlying body shape from a sequence of 3D clothed human scans.

Inner-layer template sstimation. First, we estimate a cross-frame inner-layer body template $\mathbf{V}^t$ for the subject, similar to Section 4.2 in [Zhang et al. 2017]. We sample 30 frames in the captured sequence, and fuse the whole-body tracked surface in rest pose $\mathbf{V}_i$ for those frames into a single shape $\mathbf{V}^{\mathrm{Fu}}$. Now we estimate $\mathbf{V}^t$ using the following properties of the fused shape $\mathbf{V}^{\mathrm{Fu}}$: (1) all the upper clothing vertices in $\mathbf{V}^{\mathrm{Fu}}$ should lie outside of the inner-layer body shape $\mathbf{V}^t$; (2) all the vertices not belonging to the upper clothing region in $\mathbf{V}^{\mathrm{Fu}}$ should be close to $\mathbf{V}^t$. Based on these observations, we solve $\mathbf{V}^t \in \mathbb{R}^{N_v \times 3}$ by the following optimization problem:

$$\min_{\mathbf{V}^t} E^t = w^t_{\mathrm{out}} E^t_{\mathrm{out}} + w^t_{\mathrm{fit}} E^t_{\mathrm{fit}} + w^t_{\mathrm{vis}} E^t_{\mathrm{vis}} + w^t_{\mathrm{cpl}} E^t_{\mathrm{cpl}} + w^t_{\mathrm{lpl}} E^t_{\mathrm{lpl}}. \quad (1)$$

In particular E^t_{out} penalizes any upper clothing vertex of $\mathbf{V}^{\mathrm{Fu}}$ that lies inside $\mathbf{V}^t$ by

$$E^t_{\mathrm{out}} = \sum_{\mathbf{v}_j \in \mathbf{V}^{\mathrm{Fu}}} s_j \min\{0, d(\mathbf{v}_j, \mathbf{V}^t)\}^2, \quad (2)$$

where $d(\cdot,\cdot)$ computes the signed distance from the vertex $\mathbf{v}_j$ to the surface $\mathbf{V}^t$, which takes a positive value if $\mathbf{v}_j$ lies outside of $\mathbf{V}^t$ and a negative value if $\mathbf{v}_j$ lies inside. s_j is the result of mesh segmentation; it takes the value of 1 if $\mathbf{v}_j$ is labeled as upper clothing, and 0 if $\mathbf{v}_j$ is otherwise labeled. To prevent the inner-layer template from being excessively thin, we also use a 'fit' term that penalizes too large distance between $\mathbf{V}^{\mathrm{Fu}}$ and $\mathbf{V}^t$ similar to [Zhang et al. 2017]:

$$E^t_{\mathrm{fit}} = \sum_{\mathbf{v}_j \in \mathbf{V}^{\mathrm{Fu}}} s_j d(\mathbf{v}_j, \mathbf{V}^t)^2, \quad (3)$$

with the weight of this term smaller than the 'out' term $w_{\mathrm{fit}} < w_{\mathrm{out}}$. Also notice that the vertices of $\mathbf{V}^{\mathrm{Fu}}$ with $s_j = 0$ should be in close proximity to the visible region of $\mathbf{V}^t$. This constraint is enforced by following 'vis' term:

$$E^t_{\mathrm{vis}} = \sum_{\mathbf{v}_j \in \mathbf{V}^{\mathrm{Fu}}} (1 - s_j) d(\mathbf{v}_j, \mathbf{V}^t)^2. \quad (4)$$

In addition, to regularize the inner-layer template, we impose a coupling term and a Laplacian term. Different from [Zhang et al. 2017], the topology of our inner-layer template is incompatible with the SMPL [Loper et al. 2015] model topology, so we cannot use the cross-identity body shape space of SMPL for regularization. Instead, our coupling term E^t_{cpl} enforces similarity between $\mathbf{V}^t$ and the whole-body template used in Section 1.2. The Laplacian term E^t_{lpl} penalizes large Laplacian values in the estimated inner-layer template $\mathbf{V}^t$. In our experiment, we use the following loss weights: $w^t_{\text{out}} = 1.0, w^t_{\text{fit}} = 0.03, w^t_{\text{vis}} = 1.0, w^t_{\text{cpl}} = 500.0, w^t_{\text{lpl}} = 10000.0$.

After running the optimization, we get an inner-layer template in the rest pose $\mathbf{V}^t$. This template represents the average body shape under the upper clothing, along with lower body shape with pants and various exposed skin region such as face, arms and hands. In the next step, we will use the template as a strong prior to estimate the frame-specific inner-layer body shape.

Per-frame inner-layer body shape estimation. Given the estimated inner-layer template in the previous step, now we individually estimate the inner-layer body shape for every frame in the sequence. For each frame, the estimated inner-layer shape combined with registered upper clothing mesh (Section 1.3) should resemble the whole-body surface $\hat{\mathbf{V}}_i$ when observed from outside, and allow us to render the full-body appearance of the person. For this purpose it is important that the estimated inner-layer shape is completely under the upper clothing mesh in the upper body region without intersection between the two layers.

For each frame i, we estimate an inner-layer shape $\mathbf{V}^{\text{In}}_i \in \mathbb{R}^{N_v \times 3}$ in the rest pose. We use the same LBS function $\mathcal{W}(\cdot,\cdot)$ as in Section 1.2 to transform $\mathbf{V}^{\text{In}}_i$ into the target pose $\hat{\mathbf{V}}^{\text{In}}_i = \mathcal{W}(\mathbf{V}^{\text{In}}_i, \boldsymbol{\theta}_i)$. We solve the following optimization problem:

$$\min_{\mathbf{V}^{\text{In}}_i} E^I = w^I_{\text{out}} E^I_{\text{out}} + w^I_{\text{vis}} E^I_{\text{vis}} + w^I_{\text{cpl}} E^I_{\text{cpl}}. \tag{5}$$

Our two-layer formulation requires that the estimated inner-layer shape stays strictly inside the upper clothing. Therefore, we introduce a minimum distance of $\varepsilon = 10$ millimeter that any vertex in the upper clothing should keep away from the inner-layer shape, and use

$$E^I_{\text{out}} = \sum_{\mathbf{v}_j \in \hat{\mathbf{V}}_i} s_j \min\{0, d(\mathbf{v}_j, \mathcal{W}(\mathbf{V}^{\text{In}}_i, \boldsymbol{\theta}_i)) + \varepsilon\}^2, \tag{6}$$

where, with a slight abuse of notation, s_j denotes the segmentation results for vertex $\mathbf{v}_j$ in the mesh $\hat{\mathbf{V}}_i$, with the value of 1 for a vertex in the upper clothing and 0 otherwise. Similarly, for directly visible regions in the inner-layer we have

$$E^I_{\text{vis}} = \sum_{\mathbf{v}_j \in \hat{\mathbf{V}}_i} (1 - s_j) d(\mathbf{v}_j, \mathcal{W}(\mathbf{V}^{\text{In}}_i, \boldsymbol{\theta}_i))^2. \tag{7}$$

We also couple the frame-specific rest-pose shape with the cross-frame inner-layer template to make use of the strong prior encoded in the template:

$$E^I_{\text{cpl}} = \|\mathbf{V}^{\text{In}}_{i,e} - \mathbf{V}^t_e\|^2, \tag{8}$$

where, similar to Eq. 5 in [Zhang et al. 2017], the subscript e denotes that the coupling is performed on the edges of the two meshes. In our experiment, we use the following loss weights: $w^I_{\text{out}} = 1.0, w^I_{\text{vis}} = 1.0, w^I_{\text{cpl}} = 500.0$.

Solving Eq. 5 gives us an estimation of inner-layer shape in a registered topology for each frame in the sequence. The inner-layer meshes and the outer-layer meshes obtained in Section 1.3 are both essential for our two-layer codec avatars.

2 IMPLEMENTATION DETAIL

In this section, we provide implementation detail of our method, including loss weights and network architecture.

2.1 Loss Weights

The loss weights below are provided with any involved length in the unit of millimeter (the Laplacian term in Eq. (1-3) and depth falloff scale in Eq. (4)).

Section 5.1 Body Decoder Eq. (1):

$$\lambda_g = 0.5, \quad \lambda_{lap} = 50.0, \quad \lambda_t = 5.0.$$

Section 5.2 Clothing Network Eq. (2):

$$\lambda_g = 0.5, \quad \lambda_{lap} = 50.0, \quad \lambda_t = 5.0, \quad \lambda_{kl} = 1.0.$$

We train the above two networks for 40k iterations with a batch size of 8, implemented together in group convolution for computation efficiency.

Section 5.3 Inverse Rendering with Two-layer Representation Eq. (3):

$$\lambda_i = 10.0, \quad \lambda_m = 1000.0, \quad \lambda_v = 0.001, \quad \lambda_{lap} = 100.0.$$

We train the network for 100k iterations with a batch size of 8.

Eq. (4):

$$c = 10.0.$$

For all the networks above, we use the AdamW optimizer with parameters $\alpha = 1 \times 10^{-3}$ (learning rate), $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$, except when we fine-tune the anchor VAE to individual chunks for photometric texture alignment (Section 5.4), where we use $\alpha = 1 \times 10^{-4}$.

2.2 Network Architecture

2.2.1 Body and Clothing Networks. We describe the network architecture for our body and clothing in Fig. 4, corresponding to the networks in Fig. 4 of the main paper. The detail of each module, including encoder, decoder and shadow network, is shown in Fig. 3, and each element in Fig. 3 is further defined in Fig. 2.

Geometry. Following previous work [Bagautdinov et al. 2021], the input and output geometry in both networks are defined in the unposed space (converted from the world space by inverse LBS with known joint angles) and with mean value subtracted, represented by a 3-channel UV map.

Texture. We also follow the practice in previous work [Bagautdinov et al. 2021] to separate the final view-conditioned texture into three components: a view-independent texture across different views, an additive residual texture to encode per-view variation, as well as a multiplicative shadow map to encoder long-range shadowing effects. The shadow map is predicted by a shadow network conditioned on an ambient occlusion map that is computed from the reconstructed body and clothing geometry. This separation is only formulated as an inductive bias in the network, and supervision is only applied to the final view-dependent texture.

Input conditioning. As explained in Sec. 5 of the main paper, our body and clothing decoders take in different input conditioning.

- The pose encoding and face encoding are computed by tiling the pose vector and facial keypoint vector respectively along

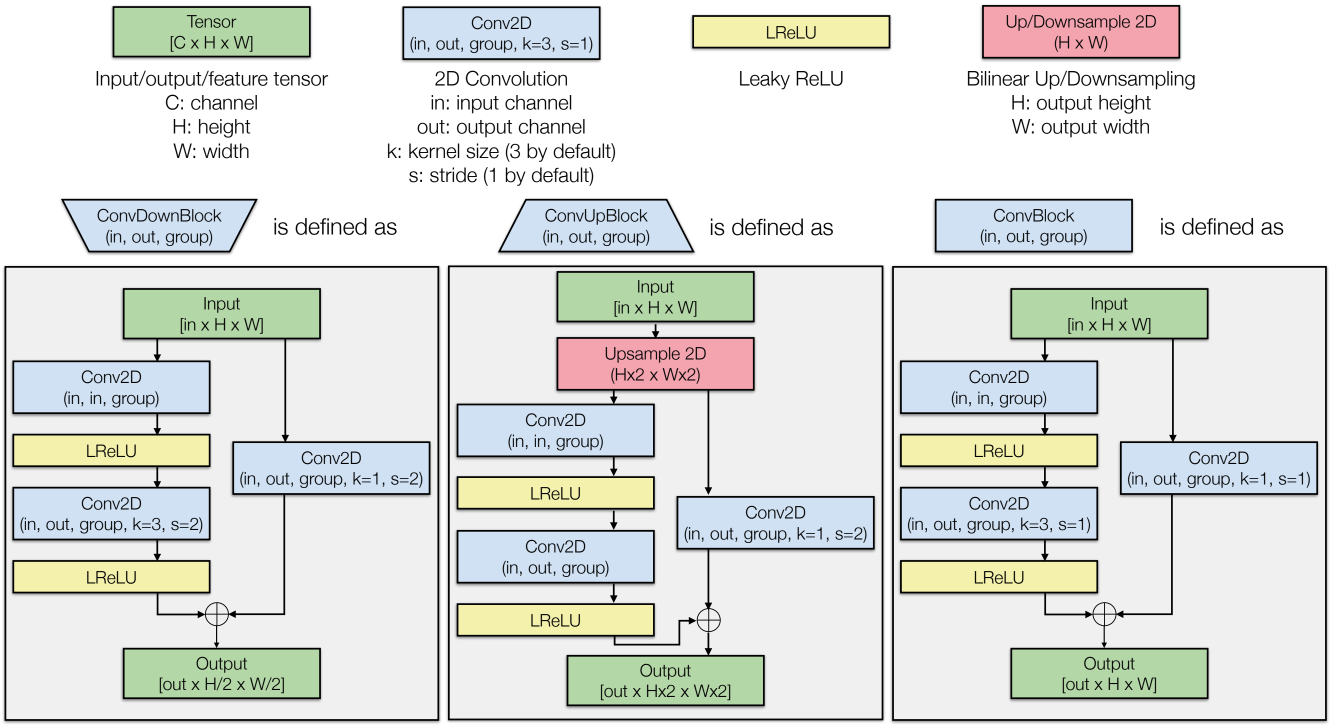

Fig. 2. Definition of network elements that will be used in the following figures for network architecture. We represent data tensor in green, convolution operations in blue, leaky ReLU in yellow, and up-sampling/down-sampling in red. We also define three types of residual convolution blocks with different up-sampling or down-sampling effects.

the spatial UV dimension to a 32×32 feature map. The feature map is then processed by a spatially localized masking operation and then a 2D residual convolution block. For the detail of this process, see Section 3.3 of [Bagautdinov et al. 2021]).

- The view conditioning is computed as the ray direction from the camera center to the reconstructed geometry in each location of the UV map, with the global orientation factored out. This 3-dim ray direction is then converted to a 64-dim feature independently for each UV location by a fully-connected layer and non-linearity (Leaky ReLU).
- The latent encoding is directly up-sampled from the latent code predicted by the encoder to keep the spatial dimension consistent with other input conditioning.

2.2.2 *Temporal Convolution Network for Clothing Animation.*

For the temporal convolution network used in Sec. 6 of the main paper, we use a total of 6 'ConvDownBlock's defined in Fig. 2, only with the 'Conv2D' operation replaced by 'Conv1D' along the temporal dimension. The input channel dimension is 94 corresponding to the pose vector, and the channel number after each 1D convolution block is 128, 128, 256, 512, 1024, 8192 respectively. The final output is reshaped to match the dimension of the clothing VAE latent code.

3 SUPPLEMENTARY VIDEO: ABLATION ANALYSIS

In the supplementary video, we show a comparison of animation results using different lengths of temporal window L as input to our TCN (Sec. 6), including $1, 3, 8, 15, 30, 60$.

We observe that using a small temporal window length (for example $L = 1, 3, 8$) leads to unnatural jittering in the animation output. Our analysis is that, as in the situation with [Bagautdinov et al. 2021], similar poses (or short pose sequences) in the training set may correspond to drastically different clothing states, and the network can overfit to the data by trying to distinguish between the nuances in pose variation, thus predicting highly different clothing states in consecutive frames. By contrast, using a larger temporal window length allows the network to take a longer history information of body motion into consideration, and thus becomes less prone to the overfitting problem. In addition, the temporal convolution architecture itself tends to predict temporally smooth output, and can also help avoid the jittering. Obviously, using a too long temporal window is also bad for model efficiency. Thus we empirically choose $L = 15$ or 30 in our model configuration.

REFERENCES

Timur Bagautdinov, Chenglei Wu, Tomas Simon, Fabian Prada, Takaaki Shiratori, Shih-En Wei, Weipeng Xu, Yaser Sheikh, and Jason Saragih. 2021. Driving-signal aware full-body avatars. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–17.
Silvano Galliani, Katrin Lasinger, and Konrad Schindler. 2015. Massively Parallel Multi-view Stereopsis by Surface Normal Diffusion. In *Proceedings of the IEEE International*

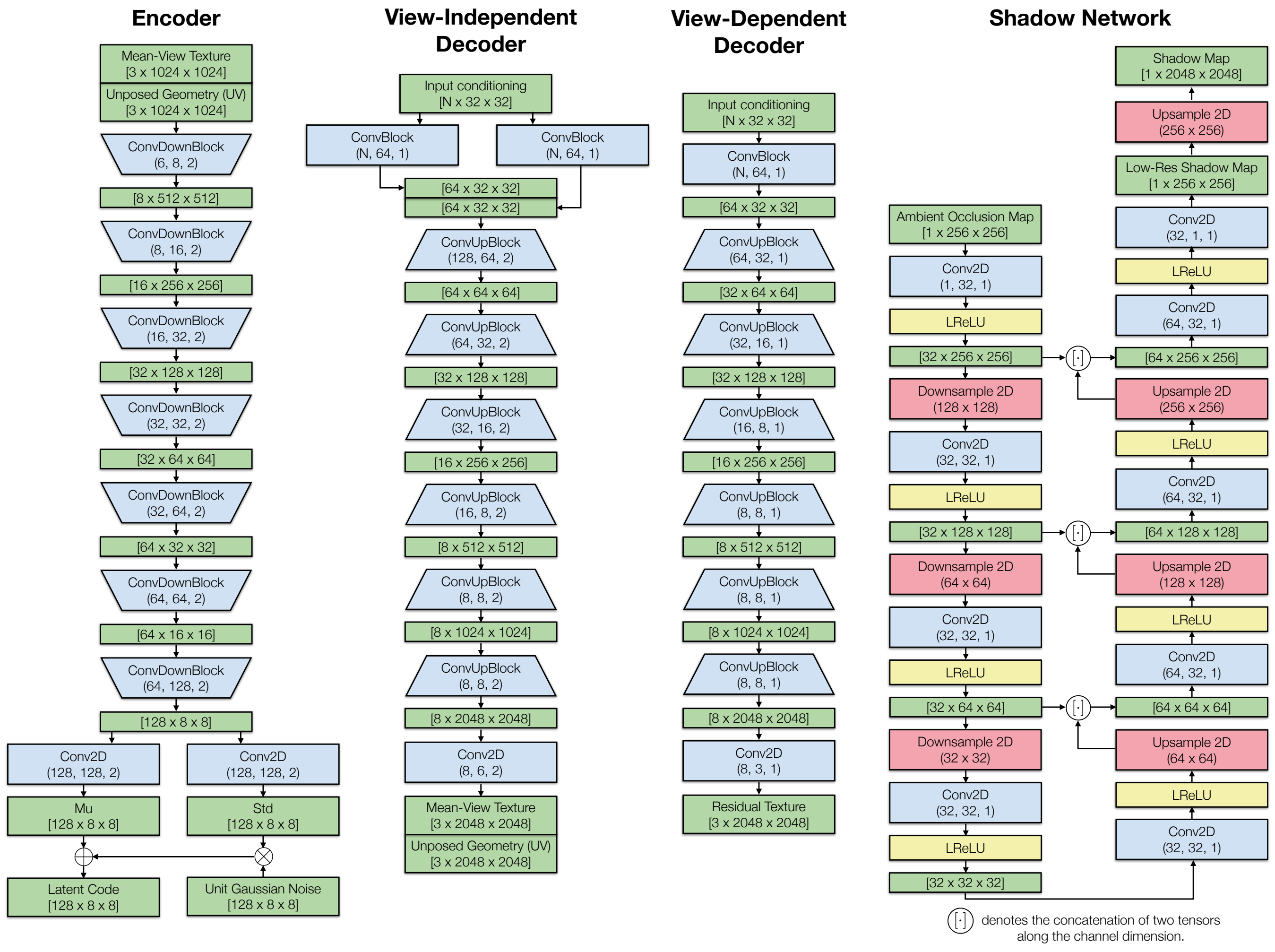

Fig. 3. Architecture of the encoder, decoder, and shadow network used in the body and clothing networks (Fig. 4). The encoder takes as input mean-view texture and unposed geometry embedded in the UV map, and outputs a latent code. The decoder consists of the two parts: the view-independent part which outputs the mean-view texture as well as geometry UV map, and the view-dependent part which outputs the per-view residual texture to be added to the mean-view texture. The shadow network has a U-Net [Ronneberger et al. 2015] architecture. It converts the Ambient Occlusion (AO) map into a shadow map.

Conference on Computer Vision (ICCV).

Alec Jacobson, Elif Tosun, Olga Sorkine, and Denis Zorin. 2010. Mixed finite elements for variational surface modeling. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 1565–1574.

Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. 2020. PointRend: Image Segmentation As Rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*. Springer, 740–755.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–16.

Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J Black. 2017. ClothCap: Seamless 4D clothing capture and retargeting. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–15.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Mingxing Tan, Ruoming Pang, and Quoc V. Le. 2020. EfficientDet: Scalable and Efficient Object Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Aaron Walsman, Weilin Wan, Tanner Schmidt, and Dieter Fox. 2017. Dynamic high resolution deformable articulated tracking. In *2017 International Conference on 3D Vision (3DV)*. IEEE, 38–47.

Chao Zhang, Sergi Pujades, Michael J. Black, and Gerard Pons-Moll. 2017. Detailed, Accurate, Human Shape Estimation From Clothed 3D Scan Sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

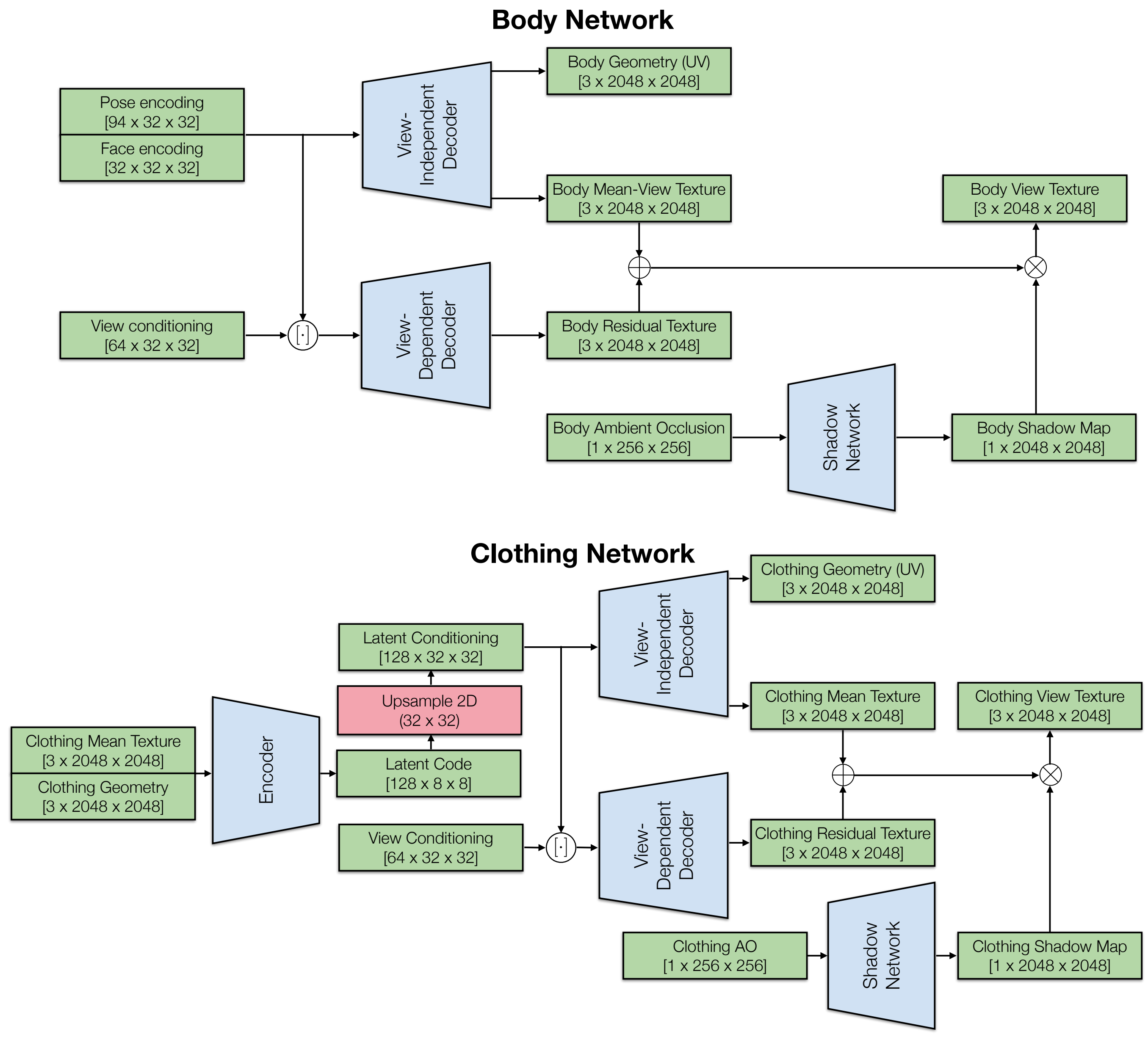
Body Network
Pose encoding [94 x 32 x 32]
Face encoding [32 x 32 x 32]
View conditioning [64 x 32 x 32]
[·]
View-Independent Decoder
View-Dependent Decoder
Body Geometry (UV) [3 x 2048 x 2048]
Body Mean-View Texture [3 x 2048 x 2048]
Body Residual Texture [3 x 2048 x 2048]
Body View Texture [3 x 2048 x 2048]
Body Ambient Occlusion [1 x 256 x 256]
Shadow Network
Body Shadow Map [1 x 2048 x 2048]
Clothing Network
Clothing Mean Texture [3 x 2048 x 2048]
Clothing Geometry [3 x 2048 x 2048]
Encoder
Latent Code [128 x 8 x 8]
Upsample 2D (32 x 32)
Latent Conditioning [128 x 32 x 32]
View Conditioning [64 x 32 x 32]
[·]
View-Independent Decoder
View-Dependent Decoder
Clothing Geometry (UV) [3 x 2048 x 2048]
Clothing Mean Texture [3 x 2048 x 2048]
Clothing Residual Texture [3 x 2048 x 2048]
Clothing View Texture [3 x 2048 x 2048]
Clothing AO [1 x 256 x 256]
Shadow Network
Clothing Shadow Map [1 x 2048 x 2048]

Fig. 4. The architecture of body and clothing networks. Notice that LBS and inverse LBS are omitted in the figure. All geometry involved is in unposed space, with mean value subtracted. The Ambient Occlusion (AO) for body and clothing is computed from the reconstructed geometry of body and clothing together.