

# TexMesh: Reconstructing Detailed Human Texture and Geometry from RGB-D Video

Tiancheng Zhi<sup>1\*</sup>, Christoph Lassner<sup>2</sup>, Tony Tung<sup>2</sup>, Carsten Stoll<sup>2</sup>,  
Srinivasa G. Narasimhan<sup>1</sup>, and Minh Vo<sup>2</sup>

<sup>1</sup> Carnegie Mellon University, {tzhi,srinivas}@cs.cmu.edu

<sup>2</sup> Facebook Reality Labs, {classner,tony.tung,carsten.stoll,minh.vo}@fb.com

**Abstract.** We present TexMesh, a novel approach to reconstruct detailed human meshes with high-resolution full-body texture from RGB-D video. TexMesh enables high quality free-viewpoint rendering of humans. Given the RGB frames, the captured environment map, and the coarse per-frame human mesh from RGB-D tracking, our method reconstructs spatiotemporally consistent and detailed per-frame meshes along with a high-resolution albedo texture. By using the incident illumination we are able to accurately estimate local surface geometry and albedo, which allows us to further use photometric constraints to adapt a synthetically trained model to real-world sequences in a *self-supervised* manner for detailed surface geometry and high-resolution texture estimation. In practice, we train our models on a short example sequence for self-adaptation and the model runs at interactive framerate afterwards. We validate TexMesh on synthetic and real-world data, and show it outperforms the state of art quantitatively and qualitatively.

**Keywords:** Human shape reconstruction, human texture generation

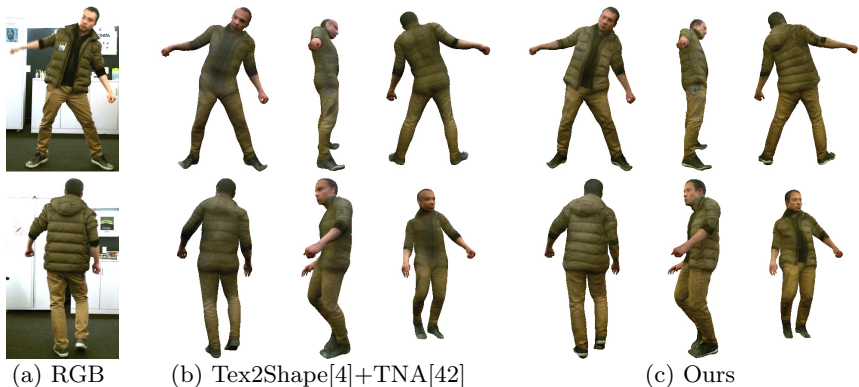
## 1 Introduction

An essential component of VR communication, modern game and movie production is the ability to reconstruct accurate and detailed human geometry with high-fidelity texture from real world data. This allows us to re-render the captured character from novel viewpoints. This is challenging even when using complex multi-camera setups [20, 49, 12, 29]. Recent works such as Tex2Shape [4] and Textured Neural Avatars [42] (TNA) have shown how to reconstruct geometry and texture respectively using nothing but a single RGB image/video as input.

Fig. 1 shows examples of novel viewpoint synthesis using Tex2Shape for geometry reconstruction and TNA for texture estimation. Tex2Shape is a single view method trained only on synthetic images without adaptation to real data. Hence, it generates the rough shape of the actor but misses some of the finer geometric details that appear in the real data (Fig. 1 (b)), and often hallucinates incorrect deformation memorized from its training data especially in occluded

---

\* Work was done during TZ internship at Facebook Reality Labs, Sausalito, CA, USA.



**Fig. 1.** Tex2Shape [4] + TNA [42] vs. Our Result. The mesh with texture is rendered from different viewpoints. Our approach reconstructs more detailed geometry, such as the moving jacket, as well as more accurate texture.

parts. As TNA does not consider the input lighting, the estimated texture contains the baked-in lighting of the original input sequence. Besides, due to small geometric misalignments, the estimated texture is blurry.

To address these issues, we introduce TexMesh, a novel framework to reconstruct both significantly higher quality mesh and texture from a real world video (see Fig. 1 (c)). Our model takes an RGB video, a corresponding environment map, and a per-frame coarse mesh as inputs, and produces a per-frame fine mesh and a high-resolution texture shared across the whole video that can be used for free-viewpoint rendering. The coarse mesh is a parametric human model obtained by 3D tracking from an RGB-D camera [51]. We use a short real video clip for self-adaptation after which TexMesh runs at 18 fps (not including human segmentation). In the pipeline, depth is used only for obtaining the coarse mesh.

Concretely, for texture generation, we parameterize the texture using a CNN and optimize it on real data by comparing the rasterized images with a limited number of selected key albedo images. Our design offers three benefits: no shading and texture mixing, less geometric misalignment leading to less blur, and built-in CNN structure prior for noise and artifact removal [48]. For mesh reconstruction, we propose to first pre-train a displacement map prediction model on synthetic images with supervision, and later optimize it on a real sequence in a self-supervised manner using photometric perceptual loss and spatiotemporal deformation priors to obtain detailed clothing wrinkles even for occluded parts.

Experiments show that the proposed method provides clear texture with high perceptual quality, and detailed dynamic mesh deformation in both the visible and occluded parts. The resulting mesh and texture can produce realistic free-viewpoint rendering (Fig. 11) and relighting results (Fig. 12).

**Contributions:** (1) We present a self-supervised framework to adapt the training on synthetic data to real data for high quality texture and mesh reconstruction. This framework is based on (2) a texture generation method including

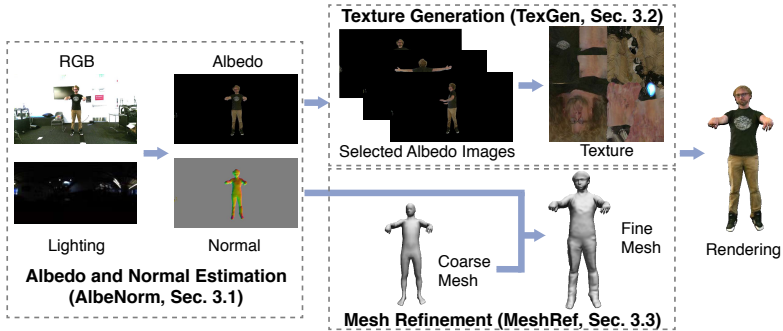
albedo estimation, frame selection, and CNN refinement and (3) a mesh reconstruction method that faithfully reconstructs clothing details even in invisible regions using shading and spatiotemporal deformation priors. Our method enables state of art free-viewpoint rendering of humans on challenging real videos.

## 2 Related Work

**Human Shape Reconstruction.** The key to human shape reconstruction is to incorporate human priors to limit the solution space. Template-based methods [15, 53] obtain human geometry by deforming the pre-scanning model. Model-based methods [60, 18, 9, 24, 37, 16, 21, 34, 54] fit a parametric naked-body model [28] to 2D poses or silhouette. While these methods estimate the coarse shape well, the recovered surface geometry is usually limited to tight clothing only [8]. To tackle this problem, [57, 56] combine depth fusion [32, 31] and human priors [28] and show highly accurate reconstruction in visible parts but not occluded regions. With multiple images, [2, 1, 3] model clothing by deforming a parametric model [28] to obtain an animatable avatar, which enables powerful VR applications. However, the clothing details are inconsistent across frames, making the re-targeting result not faithful to the observation. Some methods treat clothing as separate meshes, providing strong possibilities for simulation, but are limited to a single clothing [23], pre-defined categories [6], or mechanical properties [58]. Recently, single image methods utilizes deep learning for recovering detailed shapes, including UV space methods [23, 4], volumetric methods [59], implicit surface [39, 17], and method combining learning and shading [61]. They provide excellent details in visible regions, but hallucinate invisible parts rather than using temporal information for faithful reconstruction. In contrast to the above methods, we exploit photometric and spatiotemporal deformation cues to obtain detailed mesh, even in occluded regions.

**Human Texture Generation.** The key of texture generation is to fuse information from multiple images. Sampling based methods [2, 3] sample colors from video frames and merge them together. TNA [42] uses photometric supervision from rendering. These methods work well for videos with limited deformation but fail when the misalignment caused by large clothing deformation is significant. Single view methods [14, 33] avoid the problem of fusing multi-view information by hallucinating the occluded part. Yet, the hallucinated texture may not match the real person. Different from these methods, our method handles large deformation and provides high quality albedo texture.

**Face Reconstruction.** Face modeling is closely related to body modeling but with limited self-occlusions. Methods using photometric cues reconstruct detailed geometry and albedo via self-supervision [45, 46]. Deep learning also provides the opportunity for learning face geometry from synthetic data [40] or both synthetic data and real data [41]. These methods achieve high quality results but cannot be trivially extended to full body, especially for occluded parts.



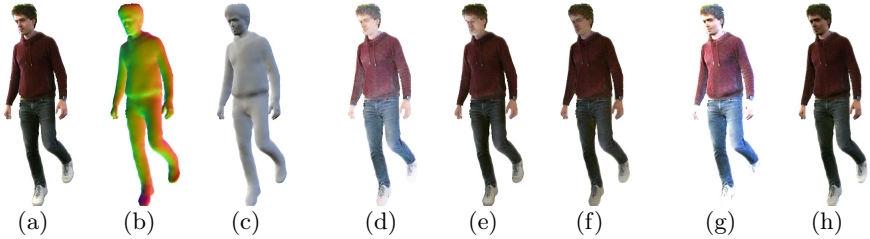
**Fig. 2.** Framework Overview. Our method consists of three modules: Albedo and Normal Estimation (AlbeNorm) pre-processes RGB images to estimate albedo and normal; Texture Generation (TexGen) selects key albedo frames and recovers a texture map; Mesh Refinement (MeshRef) takes a coarse mesh and a normal image as input and outputs a fine mesh. We pre-train AlbeNorm and MeshRef on synthetic data. Then given a short clip of the real video, we optimize TexGen to obtain texture and finetune MeshRef via self-supervision. Finally, we run AlbeNorm and MeshRef on the whole video for fine meshes.

### 3 Method

As in Fig. 2, our framework consists of three modules: Albedo and Normal Estimation (AlbeNorm), Texture Generation (TexGen), and Mesh Refinement (MeshRef). AlbeNorm takes an RGB image and the lighting, represented using Spherical Harmonics [36], and estimates texture and geometry information in the form of albedo and normal images. This is used consecutively to refine the texture and geometry estimates: TexGen selects albedo key frames and generates a high-resolution texture map from them. MeshRef takes a coarse mesh from RGB-D tracking and a normal image and estimates a refined mesh. Ground truth data for these tasks is naturally scarce. However, we observe that (1) synthetic data can be used to train the AlbeNorm and MeshRef. In synthetic settings we can use datasets with detailed person models to obtain highly detailed geometry estimates; (2) TexGen and MeshRef can be finetuned on a short sequence using perceptual photometric loss and the spatiotemporal deformation priors in a *self-supervised* manner. This makes training on large annotated video datasets obsolete. While we train AlbeNorm using only synthetic data, the model empirically generalizes well to real data. The final results are a single high-resolution full-body texture for the whole sequence and fine body geometry predicted at every frame. We describe our method in details in the following sections.

#### 3.1 Albedo and Normal Estimation

The cornerstone for our method is a good albedo and normal estimation: the normal is key to recover detailed geometry in MeshRef and the albedo is key to



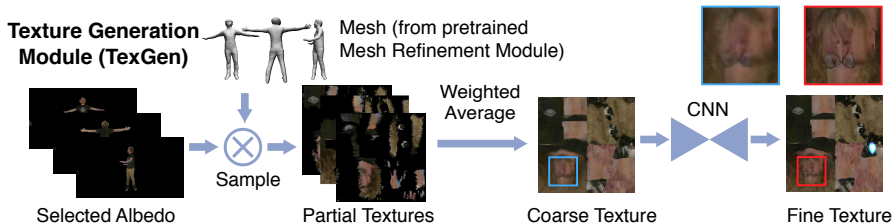
**Fig. 3.** Intermediate results of AlbeNorm. (a) original RGB image; (b) predicted normal; (c) calculated shading; (d) albedo directly from CNN; (e) albedo from dividing RGB by shading; (f) final albedo; (g) rendering using (d); (h) rendering using (f). The final albedo (f) includes less shading information than (e) (e.g., the face region), and (h) resembles the original RGB (a) better than (g).

estimate clear texture in TexGen. To extract albedo and normals, under the usual assumptions of Lambertian materials, distant light sources, and no cast shadows, we can fully represent the geometry and color of an image using a normal image and an albedo image. The normal encodes the local geometry information, and together with the incident illumination, it can be used to generate a shading image. The albedo encodes the local color and texture. The decomposition into shading and albedo is typically not unique, as we can potentially explain texture changes through normal changes. This is where the AlbeNorm module comes into play: to prevent shading from ‘leaking’ into texture and the albedo gradients from being explained as geometry change, we use the module to decouple the two components. Unlike [2], we resolve the scale ambiguity with the known lighting.

The AlbeNorm module uses a CNN to predict albedo and normal images. The inputs to this module are a segmented human image [11, 35] and the incident illumination represented as Spherical Harmonics [36]. Knowing the lighting information, we omit the scene background and process the masked human region. Concretely, let  $A_p$  and  $A_g$  be the predicted and ground truth albedo images,  $N_p$  and  $N_g$  be the predicted and ground truth normal images, and  $M$  the human mask, respectively. Then, our supervised loss  $L_{AN}$  with weights  $\lambda_a^{an}$  and  $\lambda_n^{an}$  is:

$$L_{AN} = \lambda_a^{an} \|(A_p - A_g) \cdot M\|_1 + \lambda_n^{an} \|(N_p - N_g) \cdot M\|_1, \quad (1)$$

To faithfully recover the color and texture for rendering applications, the albedo and normal should be consistent. In other words, the image synthesized using the albedo and normal should match the original image. However, as shown in Fig. 3, due to the domain gap between real and synthetic data, the synthesized image (g) does not have a similar appearance as the original one (a). Another way to obtain consistent albedo is to use the normal  $N_p$  and the input illumination to estimate the shading [36] (c), and estimate the albedo (e) by dividing the image (a) by this normal estimated shading. This albedo (e) is consistent with the



**Fig. 4.** Texture Generation Module (TexGen). TexGen selects  $K$  albedo images and converts them into partial textures. A coarse full texture is constructed by averaging the partial textures with weights. The texture is then refined by a CNN optimized from scratch for each video. The supervision comes from rasterizing the texture to the image space and comparing it with the input albedo images.

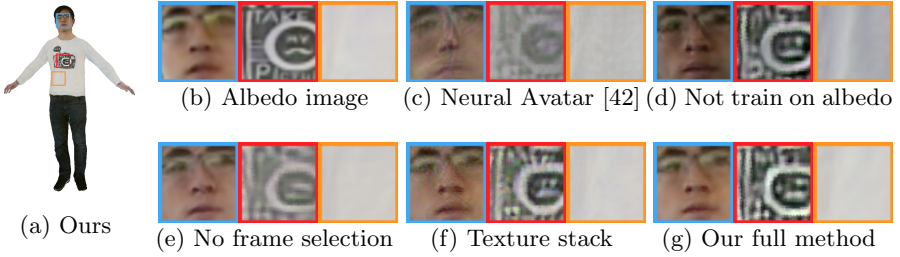
estimated normal  $N_p$ , and thus has the correct color and global scale. Yet it does not have a “flat” appearance, which means there is residual shading information included due to incorrectly estimated normals. The estimated albedo  $A_p$  in (d) on the other hand correctly factored out the shading, but is not consistent with the normal image. To consolidate the two estimates, we modify (d) by taking the color and scale from (e), to obtain an albedo (f) which is consistent with the normal image and at the same time has a “flat” appearance.

Concretely, let  $I$  be the per-pixel intensity of R,G,B channels:  $I = (R + G + B)/3$ , and  $med(I)$  be the median intensity within human mask. We first take the color from (e) as  $R' = I_d/I_e \times R_e$  and globally scale it to (e) as  $R = med(I_e)/med(I') \times R'$ .  $B$  and  $G$  are obtained similarly. The resulting albedo (f) is consistent with the normal image (b) and the newly synthesized image (h) better matches the original image (a).

### 3.2 Texture Generation

TexGen is used to encode and refine the person texture. Inspired by TNA [42], the texture is obtained by optimizing the photometric loss between rendered images and the original ones. We propose to (1) use albedo instead of the original image to prevent shading leaking into texture, (2) select keyframes to mitigate geometric misalignment, and (3) use a CNN to parameterize the texture to reduce noise and artifacts. We assume the availability of MeshRef (Sec. 3.3) pre-trained on synthetic data. Fig. 4 shows the TexGen pipeline.

**UV Mapping.** We follow the common practice in graphics where texture is stored using a UV map [7]. This map unwraps a mesh into 2D space, called UV space. Each pixel  $\mathbf{t} = (u, v)$  in UV space corresponds to a point  $\mathbf{v}$  on the mesh, and thus maps to a pixel  $\mathbf{p} = (x_{2D}, y_{2D})$  in the image via projection. Therefore, we can sample image features and convert them into UV space. For example, we can convert an albedo image into a partial texture via sampling and masking with visibility. See supplementary material for details.

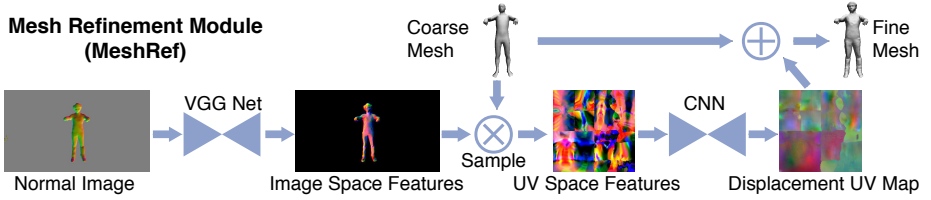


**Fig. 5.** Rasterized albedo using generated texture. (b) is the albedo from AlbeNorm, which can be seen as “ground truth”. Training on the original image rather than albedo (d) causes the texture to include shading. No frame selection (e) makes the result blurry. Using a texture stack instead of a CNN (f) creates a noisy face. (c) is TNA [42] (trained on original images using a texture stack, without frame selection). These issues are addressed by our full method (a)(g).

**Key Frame Selection.** We aim to extract a sharp texture from a short video sequence. Inherently, this is difficult because of misalignments. We aim to address these issues through selection of a few, particularly well suited frames for this task. Our selection should cover the whole body using a small number ( $K$ ) of albedo frames based on the visibility of the partial texture image. More concretely, let  $V_i$  be the visibility UV map for the  $i$ -th selected frame and  $V_0$  be the visibility map of the rest pose mesh. Since most salient parts (e.g., faces) are visible in the rest pose, we first select the frame closest to the rest pose by minimizing  $\|V_1 - V_0\|_1$ . We then greedily add the  $i$ -th frame by maximizing the total visibility  $\|\max_{j=1}^i V_j\|_1$ , until  $K$  frames are selected. We also assign a sampling frequency weight of  $W_i = 1/K + w_i / \sum_{i=1}^K w_i$ , where  $w_i = \|V_i - \max_{j=1}^{i-1} V_j\|_1$ , to every  $i$ -th frame. These weights bias the training to more visible frames and speed up the convergence. In practice, we add two adjacent frames with  $w = w_i/2$  of the  $i$ -th frame for denoising. Fig. 5 shows the benefit of our selection scheme which leads to more detailed and accurate reconstructions.

**Texture Refinement CNN.** From the key albedo frames, we generate partial textures and obtain a coarse texture using a weighted average, where the weight is  $w_i V_i$ . The coarse texture is processed by a CNN for generating a fine texture, using a deep image prior [48] for denoising and artifact removal (see faces in Fig. 5 (f) and (g) for the benefit of our CNN-based texture parameterization over the texture stack of TNA [42]). The loss comes from rasterizing the texture to the image space and comparing it with the selected albedo images. The gradients are also back-propagated to the mesh and thus MeshRef for a better alignment. This mesh adjustment scheme is a crucial component of the TexGen module.

Concretely, let  $R$  be the albedo image rasterized using SoftRas [27],  $A$  be the albedo image from AlbeNorm, and  $M$  be the human mask from segmentation [11, 35]. We use an  $L_1$  photometric loss and a perceptual loss [19] to compare  $R$  and  $A$  within  $M$ . We further regularize the mesh deformation by an  $L_1$  loss between



**Fig. 6.** Mesh Refinement Pipeline (MeshRef). MeshRef first extracts features from a normal image, then convert those features to UV space. The UV space features are then sent to a CNN to predict a 3D displacement map. We obtain the fine mesh by adding the displacements to the coarse mesh. This module is first trained using synthetic data with ground truth and later self-adapted on a short real sequence via photometric and temporal losses.

the Laplacian coordinates [44] of the current vertices and the vertices from the initial pre-trained MeshRef model. Our total loss  $L_{TG}$  is written as:

$$L_{TG} = \lambda_{L1}^{tg} ||(R - A) \cdot M||_1 + \lambda_{pct}^{tg} l_{pct}(R \cdot M, A \cdot M) + \lambda_{lap}^{tg} \sum_{i \in V} ||\mathbf{v}'_{p,i} - \mathbf{v}'_{o,i}||_1, \quad (2)$$

where  $V$  is the vertex index set and  $\mathbf{v}'_{p,i}$  and  $\mathbf{v}'_{o,i}$  are the Laplacian coordinates of the  $i$ -th predicted vertex and original vertex, respectively.  $l_{pct}^{tg}(x, y)$  computes an adaptive robust loss function [5] over the VGG features [43] as perceptual loss [19].  $\lambda_{L1}^{tg}$ ,  $\lambda_{pct}^{tg}$ , and  $\lambda_{lap}^{tg}$  are weights. In practice, we empirically limit the deformation of small structures such as the head and hands by locally disabling gradients, because of possible large mesh registration errors in these regions.

### 3.3 Mesh Refinement

The MeshRef module is used to refine the coarse mesh. Fig. 6 gives an overview of its design. Inspired by the effectiveness of predicting human shape deformation in UV space [4], MeshRef converts the image features into UV space to predict 3D displacement vectors. Our design takes the normal map from AlbeNorm and extracts VGG features [43] to obtain a better encoding, before converting the features to UV space. The features can be further augmented by other information such as vertex positions (see supplementary material for details).

To learn human shape priors, we pre-train MeshRef on a synthetic dataset with supervision. However, due to the domain gap, the pre-trained model does not perform well on real data. Thus, after obtaining the texture from TexGen, we adapt MeshRef on a real sequence using a photometric loss between the textured mesh and the original image. We also apply a motion prior loss [50] to enhance short-term temporal consistency. Since these losses cannot provide tight supervision for invisible regions, we further use a deformation loss to propagate the deformation from frames where those regions are visible to frames where they are not. This model is trained on batches of consecutive video frames.



**Supervised Training on Synthetic Images.** We supervise the 3D displacement maps using  $L_1$  and SSIM losses and regularize the 3D vertices using a Laplacian loss. Let  $D_p$  and  $D_g$  be the predicted and ground truth displacement maps and  $DSSIM = (1 - SSIM)/2$  be the structural dissimilarity function [52]. Our loss  $L_{MR1}$  is defined as:

$$L_{MR1} = \lambda_{L1}^{mr1} \|D_p - D_g\|_1 + \lambda_{ssim}^{mr1} DSSIM(D_p, D_g) + \lambda_{lap}^{mr1} \sum_{i \in V} \|\mathbf{v}'_{p,i} - \mathbf{v}'_{g,i}\|_1, \quad (3)$$

where  $\mathbf{v}'_{p,i}$  and  $\mathbf{v}'_{g,i}$  are Laplacian coordinates defined similar to Eq. 2, and  $\lambda_{L1}^{mr1}$ ,  $\lambda_{ssim}^{mr1}$ , and  $\lambda_{lap}^{mr1}$  are the weights between different losses.

**Self-supervised Training on Real Video Data.** For self-supervised training, we render the images using the SoftRas differentiable renderer [27] and compare with the original images. Our self-supervised loss is defined as:

$$L_{MR2} = \lambda_{pct}^{mr2} L_{pct} + \lambda_{sil}^{mr2} L_{sil} + \lambda_{temp}^{mr2} L_{temp} + \lambda_{pos}^{mr2} L_{pos} + \lambda_{lap}^{mr2} L_{lap} + \lambda_{deform}^{mr2} L_{deform}, \quad (4)$$

where  $L_{pct}$ ,  $L_{sil}$ ,  $L_{temp}$ ,  $L_{pos}$ ,  $L_{lap}$ ,  $L_{deform}$  are the perceptual loss, the silhouette loss, the motion consistency loss, the vertex position loss, Laplacian loss, deformation propagation loss, and  $\lambda_{pct}^{mr2}$ ,  $\lambda_{sil}^{mr2}$ ,  $\lambda_{temp}^{mr2}$ ,  $\lambda_{pos}^{mr2}$ ,  $\lambda_{lap}^{mr2}$ ,  $\lambda_{deform}^{mr2}$  are their corresponding weights, respectively. We introduce the losses below. For simplicity, we present the losses for one frame, omitting the summation over all frames.

**Perceptual Loss.** Let  $R$  be the rendered image,  $I$  be the original image,  $M_R$  be the rasterized silhouette and  $M_I$  be the segmented human mask, and  $M = M_R \cdot M_I$ . The loss is defined as  $L_{pct} = l_{pct}(R \cdot M, I \cdot M)$ , where  $l_{pct}$  is the robust perceptual loss function [5, 19].

**Silhouette Loss.** This loss compares the rasterized silhouette and the segmented human mask is defined as  $L_{sil} = \|(M_R - M_I) \cdot C\|_1$ , where  $C$  is the confidence map given by the segmentation algorithm [11, 35].

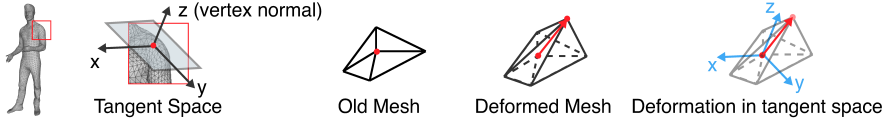
**Motion Consistency Loss.** Let  $t$  be the current frame index and  $\mathbf{v}_{p,i}^{(t)}$  be the position of the  $i$ -th vertex in frame  $t$ . Our motion consistency loss favors constant velocity in adjacent frames [50] and is written as  $L_{temp} = \sum_{i \in V} \|\mathbf{v}_{p,i}^{(t-1)} + \mathbf{v}_{p,i}^{(t+1)} -$

$2\mathbf{v}_{p,i}^{(t)}\|_1$ , where  $V$  is the set of vertex indices.

**Vertex Position Loss.** This loss prevents large deformation from the original position predicted by the model pre-trained on synthetic data and is defined as  $L_{pos} = \sum_{i \in V'} \|\mathbf{v}_{p,i} - \mathbf{v}_{o,i}\|_2$ , where  $\mathbf{v}_{p,i}$  and  $\mathbf{v}_{o,i}$  are the positions of the  $i$ -th

predicted vertex and original vertex, and  $V'$  be the set of visible vertex indices.

**Laplacian Loss.** This loss is not only applied to visible vertices but also head and hand vertices regardless of their visibility because noisy deformation of these vertices can significantly affect the perceptual result, and is defined as  $L_{lap} = \sum_{i \in V} \|(\mathbf{v}'_{p,i} - \mathbf{v}'_{o,i}) \cdot \mathbf{u}_i\|_1$ , where  $\mathbf{v}'_{p,i}$  and  $\mathbf{v}'_{o,i}$  are the Laplacian coordinates of the  $i$ -th predicted and original vertices, and  $\mathbf{u}_i$  be the weight of the  $i$ -th vertex. We set  $\mathbf{u}_i = 100$  for head and hand, 1 for other visible vertices, and 0 for the rest.



**Fig. 7.** Deformation in tangent space. At a local point, the  $z$ -axis points to the vertex normal direction and the  $x$  and  $y$  axes complete the orthogonal basis, forming a local coordinate system. We use this coordinate system to represent the vertex deformation. This representation is invariant to pose change, propagating deformation of the same vertex in different frames.

**Deformation Propagation Loss.** To reconstruct an vertex invisible in the current frame, we find a visible counterpart in the set of keyframes computed in Sec. 3.2 and propagate the deformation from it. This is similar in spirit to the canonical shape model [31]. However, because the human in the source frame and target frame may have different poses, we can not simply force the deformation in the global coordinates to be similar. We adopt the local tangent space [25] (Fig. 7) to solve this problem.

Let  $\mathbf{d}_i^{(s)}$  and  $\mathbf{d}_i^{(t)}$  be the deformation in tangent space of the  $i$ -th source vertex visible in one of the selected keyframes and occluded target vertex at the current frame. The deformation loss is defined as  $L_{deform} = \sum_{i \in V''} \|\mathbf{d}_i^{(t)} - \mathbf{d}_i^{(s)}\|_1$ ,

where  $V''$  is the set of invisible vertex indices in target frame.  $\mathbf{d}_i^{(s)}$  does not receive gradients, and is stored in a buffer updated when the source frame is sampled during training. In practice, we extend  $V''$  to include head vertices, to enhance head rigidity. Our deformation propagation scheme provides more realistic details on invisible surfaces as shown in Fig. 8.



**Fig. 8.** Effect of deformation propagation. (a) is a source frame propagating deformation to frame (b) where the back is not visible. The back of (b) is reconstructed without (c) and with (d) deformation propagation. The one with propagation shows more clothing details.

## 4 Experimental Analysis

### 4.1 Implementation Details

**Deep Networks.** CNNs are based on U-Net [38], optimized using Adam [22]. The full-frame image resolution is  $960 \times 540$  with the human region resolution around  $200 \times 430$ . Texture resolution is  $512 \times 512$ . Details are in supplementary.

**Human Model.** The full-body human model is a variant of SMPL [28]. The original SMPL model has about 7k vertices, which is too large for the coarse mesh representation, and insufficient to represent fine details such as clothing wrinkles. Thus, we construct a body model with two levels of resolution: the coarse level has 1831 vertices and 3658 faces, and the fine level has 29,290 vertices and 58,576 faces obtained by subdividing the coarse mesh topology. The vertices of the coarse mesh are a subset of the vertices of the fine mesh and share identical vertex indices. Both representations also share a unique skeletal rig that contains 74 joints. This design reduces the overhead for generating the coarse mesh, and preserves the fine mesh capability to represent geometric details.

### 4.2 Datasets

Our method requires lighting information, which is not provided by most public datasets. Thus we capture and render our own data to perform experiments.

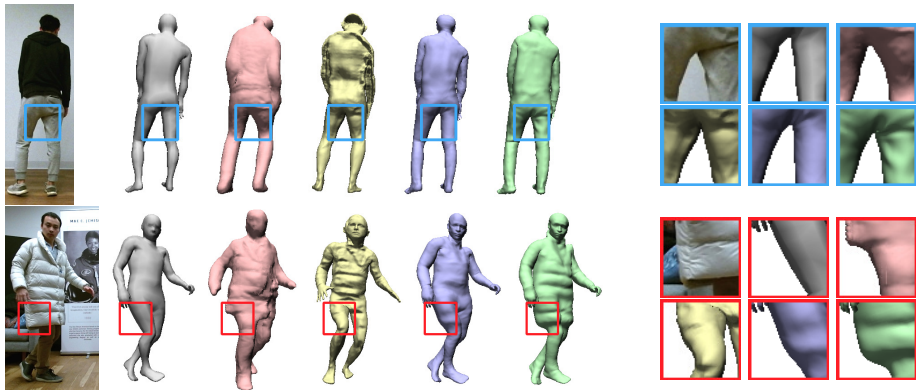
**Synthetic Images for Pre-training.** We synthesize 18,282 images using 428 human 3D scans from RenderPeople<sup>3</sup> and Our Dataset under the lighting from Laval Dataset [13]. Our Dataset was captured with a 3dMD scanner and contains 48 subjects. We registered the fine level Human Model to the 3D scans using non-rigid ICP [26], initialized with a 3D pose estimator [9]. To generate the coarse mesh, Gaussian noise scaled by a random factor sampled from uniform distribution is added to the pose and shape parameters, and the position of the character. The registered model can be set in arbitrary pose with skeletal rig. We render the 3D scans into images of various poses sampled from the Motion Capture data. No video sequences are synthesized due to its high computational demand. Our final dataset contains coarse meshes, fine meshes, displacement maps, environment maps, RGB images, albedos, normals, and human masks.

**Synthetic Videos for Quantitative Evaluation.** We synthesize 6 videos with ground truth measurements that contain dynamic clothing deformation for higher realism. Our clothing is modeled as separate meshes on top of human body scans as in DeepWrinkles [23]. However, we obtain deformation by physics-based simulation. We use the human bodies from AXYZ dataset<sup>4</sup> and the lighting from HDRI Heaven<sup>5</sup>. The videos represent subjects performing different motions such as walking or dancing and has about 3.8k frames each. In each video, we use about half of the frames for model adaptation and do inference on the rest. We treat the naked body as coarse mesh and the clothed body as fine mesh.

<sup>3</sup> <http://renderpeople.com/>

<sup>4</sup> <http://secure.axyz-design.com/>

<sup>5</sup> <http://hdrihaven.com/>



**Fig. 9.** Comparing mesh with coarse mesh (grey), DeepHuman [59] (red), HMD [61] (yellow), Tex2Shape [4]. Our method (green) outperforms them in shape and details: DeepHuman is coarse in head and hands. HMD has artifacts in head and body regions. Tex2Shape does not obtain realistic wrinkles.

**Real Videos for Qualitative Evaluation.** We capture 8 videos ( $\sim 4$ min each) using a Kinect along with lighting captured using a ThetaS. The cameras are geometrically and radiometrically calibrated. We use the first 2k frames for model adaption and infer on the whole video. We obtain the coarse mesh in real-time by solving an inverse kinematic problem to fit the posed body shape to the 3D point cloud and detected body keypoints [10] using an approach similar to [51].

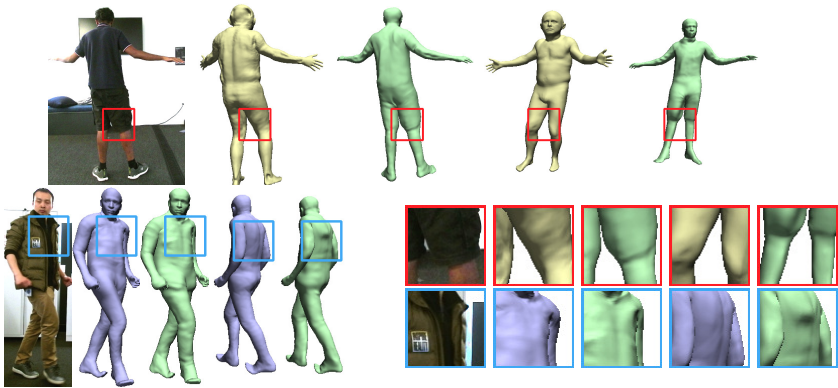
### 4.3 Results

**Texture.** We compare our texture with a sampling-based method (SBM) [3] variant and a Textured Neural Avatars (TNA) [42] variant re-implemented with our Human Model, which map between image and UV spaces using the mesh. We render albedo images on synthetic videos, and evaluate average RMSE within valid mask and MS-SSIM [52] within human bounding box over subsampled videos. Our method outperforms SBM and TNA on (RMSE, MS-SSIM): (0.124, 0.800) for SBM, (0.146, 0.809) for TNA, and **(0.119, 0.831)** for ours, respectively. See Fig. 5 and the supplementary material for qualitative results.

**Mesh.** We compare our method with DeepHuman [59], HMD [61], and a variant of Tex2Shape [4] trained on our synthetic images for our Human Model, predicting 3D displacements on top of the posed coarse mesh, using our network and loss settings. For fairness, we compare with both the original version and the variants of DeepHuman and HMD where the initial mesh is replaced by our coarse mesh. See the supplementary material about how to resolve the mismatch of camera parameters and human models. In Fig. 9, while DeepHuman and HMD provide unrealistic heads and Tex2Shape fails to produce faithful clothing details, our method is shape-preserving and generates better fine geometry. We also recover the geometry of shorts and jacket in occluded regions (Fig. 10). The supplement-

**Table 1.** Evaluation of mesh reconstruction. Silhouette IoU, rasterized normal RMSE and MS-SSIM are listed. Our method significantly outperforms the compared methods. The ablation study shows the key designs are crucial

| Method                             | IoU          | RMSE         | MS-SSIM      |
|------------------------------------|--------------|--------------|--------------|
| DeepHuman [59]                     | 0.650        | 0.399        | 0.421        |
| DeepHuman [59] variant             | 0.779        | 0.309        | 0.587        |
| HMD [61]                           | 0.667        | 0.417        | 0.684        |
| HMD [61] variant                   | 0.790        | 0.344        | 0.779        |
| Tex2Shape [4] variant              | 0.926        | 0.192        | 0.857        |
| Ours (no fine-tuning on video)     | 0.940        | 0.186        | 0.857        |
| Ours (replace input normal by RGB) | 0.928        | 0.190        | 0.852        |
| Ours (no VGG feature)              | 0.932        | 0.185        | 0.865        |
| Ours (no deformation propagation)  | <b>0.941</b> | 0.174        | 0.869        |
| Our full method                    | <b>0.941</b> | <b>0.173</b> | <b>0.870</b> |



**Fig. 10.** Viewing from front and back. Tex2Shape (blue), HMD (yellow), Ours (green). Ours captures the shape of occluded jacket hood and shorts deformation.

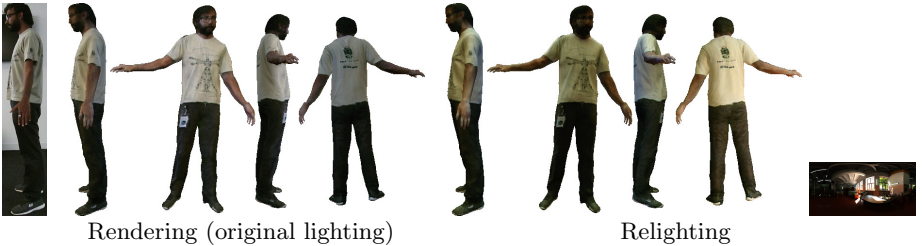
tal video shows our result is temporal consistent. Quantitatively, we evaluate on the synthetic videos by rasterizing 2D normal images. The metrics are silhouette IoU, RMSE, and MS-SSIM[52] within human mask/bounding box. Tab. 1 shows that our method outperforms the others on all metrics.

**Ablation Study.** We quantify the effect of domain finetuning, replacing the normal image by RGB image in MeshRef, removing the VGGNet, and removing the deformation propagation scheme in Tab. 1. Evidently, the first three components are crucial and ignoring them hurts the performance. As expected, removing deformation propagation has little effect because it focuses mainly on the occluded regions (see Fig. 8 for its qualitative effect).

**Applications.** We show rendering from novel viewpoints (Fig. 11) and relighting in a different environment (Fig. 12) using our outputs. The results have clear textures and realistic shading variation around clothing wrinkles.



**Fig. 11.** Rendering results. The right-most scene is from synthetic video. The rendering has both high fidelity and perceptual quality, from different viewpoints. The clothing wrinkles, logo, and text are clearly recovered.



**Fig. 12.** Free-viewpoint rendering and relighting. The detailed pattern on the shirt is clearly reconstructed. The shading varies in the clothing wrinkles implying that the wrinkles are correctly estimated as geometry rather than texture.

## 5 Conclusion

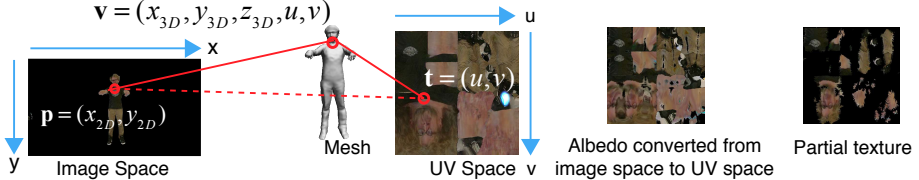
In summary, we present TexMesh, a state of art method for high fidelity human texture and geometry reconstruction, enabling high quality free-viewpoint human rendering on real videos. Our key idea is a self-supervised learning framework to adapt a synthetically-trained model to real videos. After adaptation, our model runs at interactive frame rate. In the future, we will train the model on many real videos concurrently to learn a generic shape prior as it could allow faster adaptation to new sequences or even require no adaptation at all.

Besides, we will address two limitations of our method in the future. First, we rely on a spherical camera to capture the lighting and represent it using only low frequency Spherical Harmonics. Exploring recent techniques on using a single narrow FOV image to estimate high frequency environmental lighting [41, 13] together with subtle lighting cues from human appearance [55] is an exciting future direction to tackle this problem. Second, we assume the deformation in invisible regions is the same or similar as in the keyframes, which is not always true. We will consider using clothing simulation [58] to alleviate this problem.

**Acknowledgements** We thank Junbang Liang, Yinghao Huang, and Nikolaos Sarafianos for their help with the synthetic training data generation.

# Supplementary Material

## 1 UV Mapping



**Fig. 13.** The mesh bridges image space and UV space. Assume point  $\mathbf{v}$  on the mesh is associated with its 3D position  $(x_{3D}, y_{3D}, z_{3D})$  and UV coordinates  $(u, v)$ . The corresponding point  $\mathbf{p} = (x_{2D}, y_{2D})$  in image space can be obtained via camera projection. It also corresponds to point  $\mathbf{t} = (u, v)$  in UV space. To convert features from image space to UV space, for each  $\mathbf{t}$ , we first obtain its 3D position  $(x_{3D}, y_{3D}, z_{3D})$  by barycentric interpolation, project it to image coordinates  $(x_{2D}, y_{2D})$ , and finally sample features from the image.

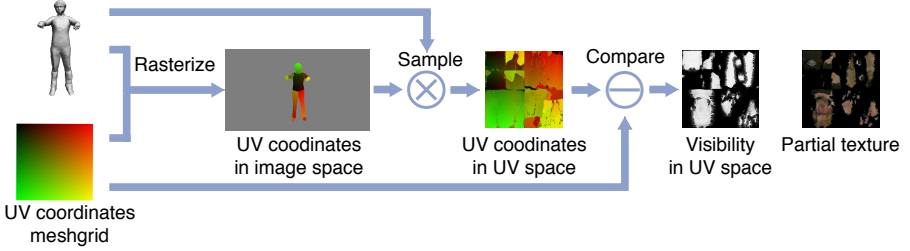
We follow the common practice in graphics where texture is stored using a UV mapping [7]. This map unwraps a mesh into 2D space, called UV space. Each pixel  $\mathbf{t} = (u, v)$  in UV space corresponds to a point  $\mathbf{v}$  on the mesh. Its 3D position is defined by the barycentric interpolation of the vertices of the face where the point is on. With the 3D position, we can project it to the image space of a calibrated camera. Thus, we can sample image features and convert them into UV space. Fig. 13 shows an example of converting albedo to UV space. It is further converted into a partial texture by masking with visibility.

## 2 Visibility Map

To calculate visibility, as shown in Fig. 14, we rasterize a image with UV coordinates, then sample that to UV space, and compare with the correct UV coordinates. The pixels whose sampled UVs are consistent with its position in UV space are visible. By masking the sampled albedo with visibility, we obtain the partial texture.

## 3 Augment MeshRef Features

In addition to VGG [43] features, we can further augment the features by including vertex position information. Specifically, we can rasterize coarse vertex position into both image space and UV space, to become a vertex position image  $I_{vp}$  and a vertex position map  $T_{vp}$ . We append  $I_{vp}$  to the input of VGGNet, and append  $T_{vp}$  to the input of the UV space CNN.



**Fig. 14.** Visibility and partial texture generation. By rasterizing UV coordinates to image space and sample it back to UV space, we obtain a UV coordinates map where only the visible parts are ”correct”. By comparing it with the ground truth UV meshgrid, we obtain the visibility map in UV space. We further calculate partial texture by masking the sampled albedo.

## 4 Implementation Details

**Network Architecture.** All the CNNs are U-Net sharing similar architectures, except for the VGG16 Network [43] in MeshRef module. See Tab. 2 for the shared components and Tab. 3, 4, and 5 for architectures of AlbeNorm, TexGen, and MeshRef CNNs. Specially, the CNN in TexGen predicts the residual between the coarse texture and the fine texture.

**Table 2.** Network Components. We use ReLU [30] for activation, and Instance Normalization [47] for normalization

| Type    | Components                                     |
|---------|--|
| inconv  | [Conv3×3 + ReLU + InstanceNorm]×2              |
| down    | [Conv3×3 + ReLU + InstanceNorm]×2 + MaxPool2×2 |
| up      | Upsample + [Conv3×3 + ReLU + InstanceNorm]×2   |
| outconv | Conv1×1  |

**Hyperparameters.** We use  $K = 30$  for the number of selected frames, and  $\lambda_a^{an} = 1, \lambda_n^{an} = 1, \lambda_{L1}^{tg} = 20, \lambda_{pct}^{tg} = 1, \lambda_{lap}^{tg} = 10, \lambda_{L1}^{mr1} = 1, \lambda_{ssim}^{mr1} = 1, \lambda_{lap}^{mr1} = 20, \lambda_{pct}^{mr2} = 1, \lambda_{sil}^{mr2} = 100, \lambda_{temp}^{mr2} = 10, \lambda_{pos}^{mr2} = 10, \lambda_{lap}^{mr2} = 10, \lambda_{deform}^{mr2} = 10$  for the loss weights. We use learning rate  $10^{-5}$  for pretraining AlbeNorm,  $10^{-4}$  for pretraining MeshRef,  $3 \times 10^{-4}$  for optimizing TexGen, and  $5 \times 10^{-5}$  for finetuning MeshRef. We use batch size 4 for pretraining AlbeNorm, 1 for pretraining MeshRef, 1 for optimizing TexGen, and 3 for finetuning MeshRef (as a triplet for motion smoothness loss). VGGNet is trained from scratch with MeshRef CNN,



**Table 3.** Network Architecture of AlbeNorm CNN

| Name                  | Type    | Input           | Output Channels |
|-----------------------|---------|-----------------|-----------------|
| inc                   | inconv  | RGB+SH lighting | 64              |
| down1                 | down    | inc             | 128             |
| down2                 | down    | down1           | 256             |
| down3                 | down    | down2           | 512             |
| down4                 | down    | down3           | 512             |
| up1a                  | up      | down4, down3    | 256             |
| up2a                  | up      | up1a, down2     | 128             |
| up3a                  | up      | up2a, down1     | 64              |
| up4a                  | up      | up3a, inc       | 64              |
| outca (Normal Output) | up      | up4a            | 3               |
| up1b                  | up      | down4, down3    | 256             |
| up2b                  | up      | up1b, down2     | 128             |
| up3b                  | up      | up2b, down1     | 64              |
| up4b                  | up      | up3b, inc       | 64              |
| outcb (Albedo Output) | outconv | up4b            | 3               |

**Table 4.** Network Architecture of TexGen CNN

| Name  | Type    | Input          | Output Channels |
|-------|---------|----------------|-----------------|
| inc   | inconv  | Coarse Texture | 64              |
| down1 | down    | inc            | 128             |
| down2 | down    | down1          | 256             |
| down3 | down    | down2          | 512             |
| down4 | down    | down3          | 512             |
| up1   | up      | down4, down3   | 256             |
| up2   | up      | up1, down2     | 128             |
| up3   | up      | up2, down1     | 64              |
| up4   | up      | up3, inc       | 64              |
| outc  | outconv | up4            | 3               |

**Table 5.** Network Architecture of MeshRef CNN. “feat0”, “feat1”, “feat2”, “feat3”, “feat4” are features converted from VGGNet input, conv1.2, conv2.2, conv3.3, and conv4.3 features

| Name  | Type    | Input        | Output Channels |
|-------|---------|--------------|-----------------|
| inc   | inconv  | feat0        | 64              |
| down1 | down    | inc, feat1   | 128             |
| down2 | down    | down1, feat2 | 256             |
| down3 | down    | down2, feat3 | 512             |
| down4 | down    | down3, feat4 | 512             |
| up1   | up      | down4, down3 | 256             |
| up2   | up      | up1, down2   | 128             |
| up3   | up      | up2, down1   | 64              |
| up4   | up      | up3, inc     | 64              |
| outc  | outconv | up4          | 3               |

and kept fixed during finetuning. To speed up finetuning, we use a smaller image size  $480 \times 270$  for photometric losses, but the image features are from the  $960 \times 540$  original image.

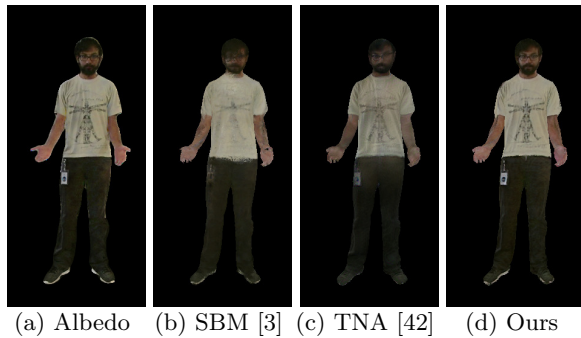
**Adaptive Robust Perceptual Loss.** We use the adaptive robust loss [5] for perceptual losses [19]. We use VGG16 conv1.2, conv2.2, conv3.3, and conv4.3 features for TexGen, and conv3.3, and conv4.3 features for MeshRef. We use learning rate  $3 \times 10^{-4}$  for the adaptive robust function.

## 5 Qualitative Texture Generation Results

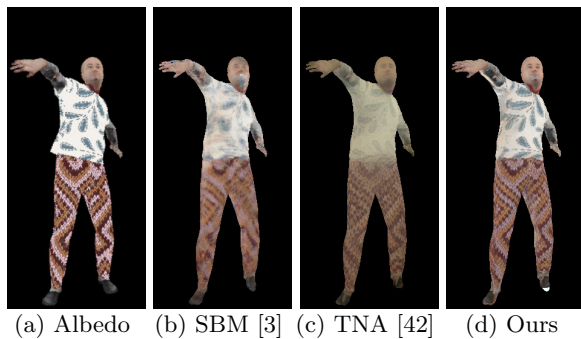
See Fig. 15 and 16 for qualitative results. Our method provides clearer texture, better contrast, and less baked-in shading.

## 6 Handle Different Human Models and Camera Parameters

We establish a mapping between SMPL [28] and Our Human Model, so that we can replace the initial meshes of DeepHuman [59] and HMD [61] by our coarse mesh. Besides, DeepHuman and HMD use different camera parameters from ours, and the cropping makes it hard to simply transform the mesh from one setting to the other. Thus, we adopt an approximate way to do this: We transform our coarse mesh to match the position and scale of their original initial mesh. Similarly, after obtaining their reconstruction result, we transform the mesh to match the position and scale of our coarse mesh, for DeepHuman, HMD, and their variants.



**Fig. 15.** Comparing texture generation by visualizing no-shading image on real data. (a) is albedo image from AlbeNorm, which can be seen as “ground truth”. Our result has clearer texture, and less baked-in shading.



**Fig. 16.** Comparing texture generation by visualizing no-shading image on synthetic data. (a) is ground truth albedo image. Our result has better contrast and clearer texture.

## References

1. Alldieck, T., Magnor, M., Bhatnagar, B.L., Theobalt, C., Pons-Moll, G.: Learning to reconstruct people in clothing from a single rgb camera. In: CVPR (2019)
2. Alldieck, T., Magnor, M., Xu, W., Theobalt, C., Pons-Moll, G.: Detailed human avatars from monocular video. In: 3DV (2018)
3. Alldieck, T., Magnor, M., Xu, W., Theobalt, C., Pons-Moll, G.: Video based reconstruction of 3d people models. In: CVPR (2018)
4. Alldieck, T., Pons-Moll, G., Theobalt, C., Magnor, M.: Tex2shape: Detailed full human body geometry from a single image. In: CVPR (2019)
5. Barron, J.T.: A general and adaptive robust loss function. In: CVPR (2019)
6. Bhatnagar, B.L., Tiwari, G., Theobalt, C., Pons-Moll, G.: Multi-garment net: Learning to dress 3d people from images. In: ICCV (2019)
7. Blinn, J.F., Newell, M.E.: Texture and reflection in computer generated images. *Communications of the ACM* **19**(10), 542–547 (1976)
8. Bogo, F., Black, M.J., Loper, M., Romero, J.: Detailed full-body reconstructions of moving people from monocular rgb-d sequences. In: ICCV (2015)
9. Bogo, F., Kanazawa, A., Lassner, C., Gehler, P., Romero, J., Black, M.J.: Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In: ECCV (2016)
10. Cao, Z., Hidalgo, G., Simon, T., Wei, S.E., Sheikh, Y.: Openpose: realtime multi-person 2d pose estimation using part affinity fields. TPAMI (2019)
11. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
12. Collet, A., Chuang, M., Sweeney, P., Gillett, D., Evseev, D., Calabrese, D., Hoppe, H., Kirk, A., Sullivan, S.: High-quality streamable free-viewpoint video. TOG (2015)
13. Gardner, M.A., Sunkavalli, K., Yumer, E., Shen, X., Gambaretto, E., Gagné, C., Lalonde, J.F.: Learning to predict indoor illumination from a single image. TOG (SIGGRAPH Asia) **9**(4) (2017)
14. Grigorev, A., Sevastopolsky, A., Vakhitov, A., Lempitsky, V.: Coordinate-based texture inpainting for pose-guided human image generation. In: CVPR (2019)
15. Habermann, M., Xu, W., Zollhoefer, M., Pons-Moll, G., Theobalt, C.: Livecap: Real-time human performance capture from monocular video. TOG (2019)
16. Huang, Y., Bogo, F., Lassner, C., Kanazawa, A., Gehler, P.V., Romero, J., Akhter, I., Black, M.J.: Towards accurate marker-less human shape and pose estimation over time. In: 3DV (2017)
17. Huang, Z., Xu, Y., Lassner, C., Li, H., Tung, T.: Arch: Animatable reconstruction of clothed humans. In: CVPR (2020)
18. Jain, A., Thormählen, T., Seidel, H.P., Theobalt, C.: Moviereshape: Tracking and reshaping of humans in videos. TOG **29**(6), 1–10 (2010)
19. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV (2016)
20. Kanade, T., Rander, P., Narayanan, P.: Virtualized reality: Constructing virtual worlds from real scenes. *IEEE multimedia* **4**(1), 34–47 (1997)
21. Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: CVPR (2018)
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)

23. Lahner, Z., Cremers, D., Tung, T.: Deepwrinkles: Accurate and realistic clothing modeling. In: ECCV (2018)
24. Lassner, C., Romero, J., Kiefel, M., Bogo, F., Black, M.J., Gehler, P.V.: Unite the people: Closing the loop between 3d and 2d human representations. In: CVPR (2017)
25. Lengyel, E.: Mathematics for 3D game programming and computer graphics. Cengage Learning (2012)
26. Li, H., Sumner, R.W., Pauly, M.: Global correspondence optimization for non-rigid registration of depth scans. In: CGF (2008)
27. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In: ICCV (2019)
28. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: Smpl: A skinned multi-person linear model. TOG **34**(6), 1–16 (2015)
29. Matsuyama, T., Takai, T.: Generation, visualization, and editing of 3d video. In: 3DPVT (2002)
30. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML (2010)
31. Newcombe, R.A., Fox, D., Seitz, S.M.: Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In: CVPR (2015)
32. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality. pp. 127–136. IEEE (2011)
33. Oechsle, M., Mescheder, L., Niemeyer, M., Strauss, T., Geiger, A.: Texture fields: Learning texture representations in function space. In: ICCV (2019)
34. Omran, M., Lassner, C., Pons-Moll, G., Gehler, P., Schiele, B.: Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In: 3DV (2018)
35. Piccardi, M.: Background subtraction techniques: a review. In: 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583). vol. 4, pp. 3099–3104. IEEE (2004)
36. Ramamoorthi, R., Hanrahan, P.: An efficient representation for irradiance environment maps. In: SIGGRAPH (2001)
37. Rhodin, H., Robertini, N., Casas, D., Richardt, C., Seidel, H.P., Theobalt, C.: General automatic human shape and motion capture using volumetric contour cues. In: ECCV (2016)
38. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI (2015)
39. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: ICCV (2019)
40. Sela, M., Richardson, E., Kimmel, R.: Unrestricted facial geometry reconstruction using image-to-image translation. In: ICCV (2017)
41. Sengupta, S., Kanazawa, A., Castillo, C.D., Jacobs, D.W.: Sfsnet: Learning shape, reflectance and illuminance of faces in the wild. In: CVPR (2018)
42. Shysheya, A., Zakharov, E., Aliev, K.A., Bashirov, R., Burkov, E., Isakov, K., Ivakhnenko, A., Malkov, Y., Pasechnik, I., Ulyanov, D., et al.: Textured neural avatars. In: CVPR (2019)
43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)

44. Sorkine, O.: Differential representations for mesh processing. In: CGF (2006)
45. Tewari, A., Bernard, F., Garrido, P., Bharaj, G., Elgharib, M., Seidel, H.P., Pérez, P., Zollhofer, M., Theobalt, C.: Fml: Face model learning from videos. In: CVPR (2019)
46. Tewari, A., Zollhöfer, M., Garrido, P., Bernard, F., Kim, H., Pérez, P., Theobalt, C.: Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In: CVPR (2018)
47. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
48. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. In: CVPR (2018)
49. Vlasic, D., Peers, P., Baran, I., Debevec, P., Popović, J., Rusinkiewicz, S., Matusik, W.: Dynamic shape capture using multi-view photometric stereo. TOG (SIGGRAPH Asia) **28**(5) (2009)
50. Vo, M., Narasimhan, S.G., Sheikh, Y.: Spatiotemporal bundle adjustment for dynamic 3d reconstruction. In: CVPR (2016)
51. Walsman, A., Wan, W., Schmidt, T., Fox, D.: Dynamic high resolution deformable articulated tracking. In: 3DV (2017)
52. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. TIP **13**(4), 600–612 (2004)
53. Xu, W., Chatterjee, A., Zollhöfer, M., Rhodin, H., Mehta, D., Seidel, H.P., Theobalt, C.: Monoperfcap: Human performance capture from monocular video. TOG (2018)
54. Xu, Y., Zhu, S.C., Tung, T.: Denserac: Joint 3d pose and shape estimation by dense render and compare. In: ICCV (2019)
55. Yi, R., Zhu, C., Tan, P., Lin, S.: Faces as lighting probes via unsupervised deep highlight extraction. In: ECCV (2018)
56. Yu, T., Guo, K., Xu, F., Dong, Y., Su, Z., Zhao, J., Li, J., Dai, Q., Liu, Y.: Bodyfusion: Real-time capture of human motion and surface geometry using a single depth camera. In: ICCV (2017)
57. Yu, T., Zheng, Z., Guo, K., Zhao, J., Dai, Q., Li, H., Pons-Moll, G., Liu, Y.: Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In: CVPR (2018)
58. Yu, T., Zheng, Z., Zhong, Y., Zhao, J., Dai, Q., Pons-Moll, G., Liu, Y.: Simulcap: Single-view human performance capture with cloth simulation. In: CVPR (2019)
59. Zheng, Z., Yu, T., Wei, Y., Dai, Q., Liu, Y.: Deephuman: 3d human reconstruction from a single image. In: ICCV (2019)
60. Zhou, S., Fu, H., Liu, L., Cohen-Or, D., Han, X.: Parametric reshaping of human bodies in images. TOG **29**(4), 1–10 (2010)
61. Zhu, H., Zuo, X., Wang, S., Cao, X., Yang, R.: Detailed human shape estimation from a single image by hierarchical mesh deformation. In: CVPR (2019)