Semantic Human Mesh Reconstruction with Textures

Supplementary Material

We provide additional information and experiments about our method. Below is a summary of the sections in the supplementary:

- Sec. G reports the implementation details of SHERT.
- Sec. H displays more experiments and results.
- Sec. I discusses the limitations.
- Sec. J shows more available applications.

G. Implementation details

G.1. SNS

Algorithm. In Alg. 1, we describe the details of semantic- and normal-based sampling.
Skinning weights. The skinning weight of a new vertex is the average of the skinning weights of the two neighboring vertices on the same edge.

G.2. Network Architecture

The architectures of our completion network and refinement network are shown in Fig. S3.

G.3. Mask Dilation in Completion

The experiments show that the vertices located at the edges of the removed triangle meshes may also be inaccurate, leading to a negative effect on the quality of the completed mesh. To tackle this problem, we dilate the hole mask, enabling the network to also fill in the surrounding areas of the eliminated triangle meshes. This leads to smoother and more realistic results.

G.4. Refinement

Feature Projection. We use camera parameters to query the pixels corresponding to the current UV coordinates in the image domain to find the corresponding image domain features (refer to Fig. S1). We will concatenate the depth of the projected vertex in camera space with the projected features.
Smoothing. We observe that when the input mesh does not align properly with the image and normal features, the predictions made by the network tend to introduce some noise, resulting in minor protrusions on the refined mesh. In order to reduce the occurrence of such distortions, we perform Laplacian Smoothing [20] on the input mesh before using it as input during inference. For the same reason, we also smooth the supervisory data in training.
Iteration. During the iteration process of the refinement network, the input image and the front-back normal maps

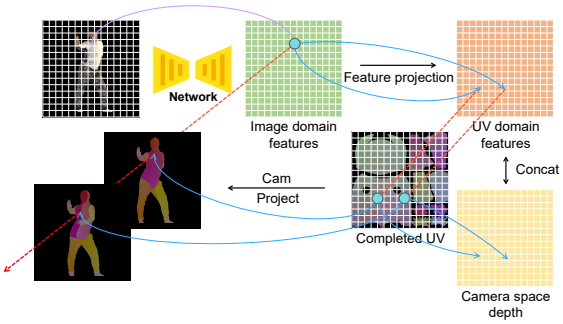

Figure S1. **Feature Projection**. Noted that the image domain features are projected to both the visible part and invisible part.

remain unchanged, and we only use the previous result generated by network as the new input.
Displacement Projection. Actually we use the projection of displacement to avoid the displacement vector being perpendicular to the normal vector. Therefore, Eq. 1 and Eq. 9 in our code should be as follows:

$$\overline{d} = \frac{(S_{sample} - S_{pose}) \cdot N_{pose}}{\|N_{pose}\|_2}, \tag{1}$$

$$\mathcal{L}_{dis} = MSE(z - \frac{(S_c - S_l) \cdot N_l}{\|N_l\|_2}). \tag{9}$$

Rendered images. We render images of THuman2.0 using orthogonal cameras. Generally, this does not consistent with the real-world images and may have an impact on performance in wild tests.

G.5. Face Substitution

FLAME to SMPL-X. We first transform the detailed faces of EMOCA [14] to the UV domain since it has more vertices than the standard FLAME [37] model. We then transform the detailed FLAME UV position map to SMPL-X [47] representation. The corresponding detailed sub-SMPLX face can be resampled from the transfered UV map. We clip the resampled face and align it with the face of sub-SMPLX, then performe vertex substitution. The whole process is shown in Fig. S2.
Face Alignment. We first rotate both the detailed face and sub-SMPLX face to align them with the positive z-axis, based on their respective average normal vectors in the x, y, and z directions. We then normalize them to a 0-1 space through translation and scaling. To achieve accurate alignment, we use the ICP algorithm [7]. Finally, we transform the aligned detailed face back into the coordinate space of the original sub-SMPLX.

Algorithm 1: Semantic- and Normal-based Sampling

Input: Target surface $\mathcal{M}_t$, sub-SMPLX $\mathcal{M}_x$, query-range r, angle-threshold t_{angle}, area-threshold t_{area}, edge-threshold t_{edge}, connectivity-threshold $t_{connect}$.
Output: Partially sampled mesh $\mathcal{M}_s$, hole mask $\mathcal{H}_o$.

Function `MeshCulling` ($\mathcal{M}_x$, $\mathcal{M}_s$, t_{angle}, t_{area}, t_{edge}) **:**
  List $invalid$;
  foreach *face* f ***in*** $\mathcal{M}_s$ **do**
    Get the corresponding face f_x of $\mathcal{M}_x$;
    Compute the normal angle $angle_{pose}$ between f and f_x;
    Compute the area ratio $area_{pose}$ between f and f_x;
    Compute the edge ratio $edge_{pose}$ between the longest and the shortest edges of f;
    if $angle_{pose} > t_{angle} \vee area_{pose} > t_{area} \vee edge_{pose} > t_{edge}$ **then**
      foreach *vertex* p ***in*** f **do**
        $invalid$.append(index(p));
  $\mathcal{M}_s \leftarrow$ DeleteFaces($invalid$);
  return $\mathcal{M}_s$;

Copy $\mathcal{M}_x$ to $\mathcal{M}_s$;
List $invalid$;
foreach *vertex* p ***in*** $\mathcal{M}_s$ **do**
  $direction \leftarrow CalNormal(p)$;
  if p *is not inside the target surface* $\mathcal{M}_t$ **then**
    $direction \leftarrow direction \cdot (-1)$;
  Find the intersection point i_p with $\mathcal{M}_t$ starting from p along $direction$ within range r;
  if i_p *does not exist* **then**
    $invalid$.append(index(p));
  else
    $p \leftarrow i_p$;
$\mathcal{M}_s \leftarrow$ DeleteFaces($invalid$);
$\mathcal{M}_s \leftarrow$ MeshCulling($\mathcal{M}_x, \mathcal{M}_s, t_{angle}, t_{area}, t_{edge}$);
$\mathcal{M}_x, \mathcal{M}_s \leftarrow$ LBSToStar($\mathcal{M}_x, \mathcal{M}_s$);
$\mathcal{M}_s \leftarrow$ MeshCulling($\mathcal{M}_x, \mathcal{M}_s, t_{angle}, t_{area}, t_{edge}$);
$\mathcal{M}_s \leftarrow$ FaceConnectivityCheckAndDelete($\mathcal{M}_s$, $t_{connect}$);
$\mathcal{H}_o \leftarrow$ GenerateHoleMask($\mathcal{M}_s$);
$\mathcal{M}_s \leftarrow$ LBSBack($\mathcal{M}_s$);
return $\mathcal{M}_s, \mathcal{H}_o$;

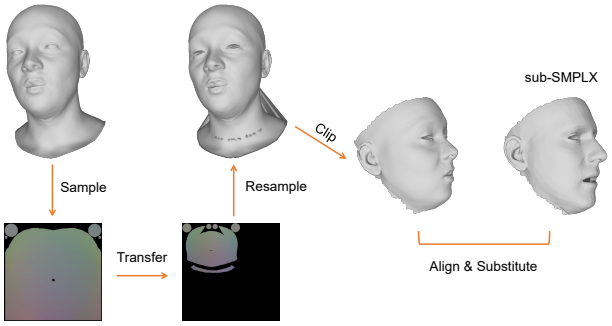

Figure S2. **Face Substitution**. We transform the detailed FLAME-based face model to sub-SMPLX representation to achieve face substitution.

Edge Smoothing. Despite already aligning the detailed face and sub-SMPLX face through affine transformations, there is still a noticeable gap at the joint of the face and head. We resample the mesh between the face and the head using the corresponding UV position map and UV mask and reduce the gap at the joint through smoothing processing. The results are shown in Fig. S4.

G.6. Texture Diffusion

Partial Texture. We obtain the partial texture by projection (in Sec. G.4). In order to prevent the front colors from projecting onto invisible areas, we send a ray from each vertex along the camera's angle of view to detect the visible set. If the ray intersects with the body mesh, the corresponding vertex is considered invisible. Additionally, we use EMOCA [14] to predict facial textures for better results.
Text Prompts. We use BLIP2 [35] to generate text prompts corresponding to textures sampled from rendered images in Thuman2.0 for diffusion training.
Inference Details. We found inaccurate results at the edges of the projection, so we conduct dilation on the projected mask. Moreover, we use the facial texture predicted by EMOCA [14] since the projected one frequently does not match the mesh. Lastly, we provide an iterative optimization scheme that allows users to manually recognize the sections to be optimized. The effects of different inpainting strategies are shown in Fig. S8. Sometimes the front and back of clothing may look similar, so we can also project the front color to the back of the mesh and conduct iterative optimization manually. This approach can achieve better results at the texture seams.

H. Additional Experiments

The influence of mask dilation. We use mask dilation to reduce the impact of the edges of the culled meshes in Sec. G.3. As shown in Fig. S5, as the degree of dilation increases, the predicted results for the holed areas and its neighboring parts become smoother, but at the same time, more geometric details belonging to the original input are

lost.

The completion results of the face, hands, and feet. The results are shown in Fig. S6

The refine iteration for textured mesh. The results are shown in Fig. S7

The influence of smoothed inputs for refinement. We demonstrate the effects of smoothed inputs for refinement in Fig. S9. The results indicate that while smoothing has minimal impact on the ultimate refinement of details, it is effective in curbing the generation of noise.

The effect of normal quality in refinement. We train the refinement network with the normal maps of THuman2.0 [70] scans, which has a resolution of 1024×1024. However, the predicted normal maps of ECON [65] are 512×512. Such difference can lead to the degradation of refinement performance. Additionally, the quality of the normal and the degree of matching with the input geometry will also affect the final results. Experiments are carried out to show such effects on refine results, as shown in Fig. S10.

Registration on incomplete and incorrect inputs. The results are shown in Fig. S11

More comparisons. We present more comparisons for monocular image reconstruction in Fig. S12. We use challenging inputs to comprehensively evaluate the performance of SHERT.

More inpainting results. We present more inpainting results on in-the-wild images in Fig. S13. The challenging inputs are also being used to demonstrate the performance unbiasedly.

I. Limitations

Incorrect sampling in SNS. SNS cannot guarantee that all of the unremoved faces are correct. When the target mesh is geometrically inseparable, SNS may incorrectly register a part of the body onto an incorrect surface.

Folded surfaces. Due to not using LBS [34] for deformation during completion, but instead using normal-based offsets, there could be some folded surfaces, leading to discrepancies with the desired outcome.

Loose clothing, hair, and feet. Due to the limited expressiveness of the SMPL-X template we used, it is difficult for SNS to fully capture the geometry of loose clothing. Additionally, SHERT cannot accurately model long hair. Lastly, replacing the feet with the SMPL-X template resulted in an inability to reconstruct normal shoes in the final result.

Smoothed details. Some details of sampling failure during SNS may be lost during completion. At the same time, the smoothing used in refinement also smoothes out a small amount of detail.

Discontinuity of texture seams. Due to texture prediction being carried out on unfolded UVs, it is difficult to ensure consistency of the content at the edges of the seams, as they are not spatially adjacent.

The corresponding results are shown in Fig. S14 and Fig. S13.

J. Applications

Mesh down-sampling. Because we have retained the original vertex ordering during the subdivision process, our representation supports direct down-sampling of the predicted results without any additional computation by using the predefined SMPL-X model (refer to Fig. S15). The low-resolution results can preserve the geometric details, material and mesh quality well.

3D face substitution. As shown in Fig. S16, SHERT enables direct 3D face substitution, allowing the face corresponding to the input image to be transferred to any existing mesh. With manual optimization, the facial texture can also be accurately transferred.

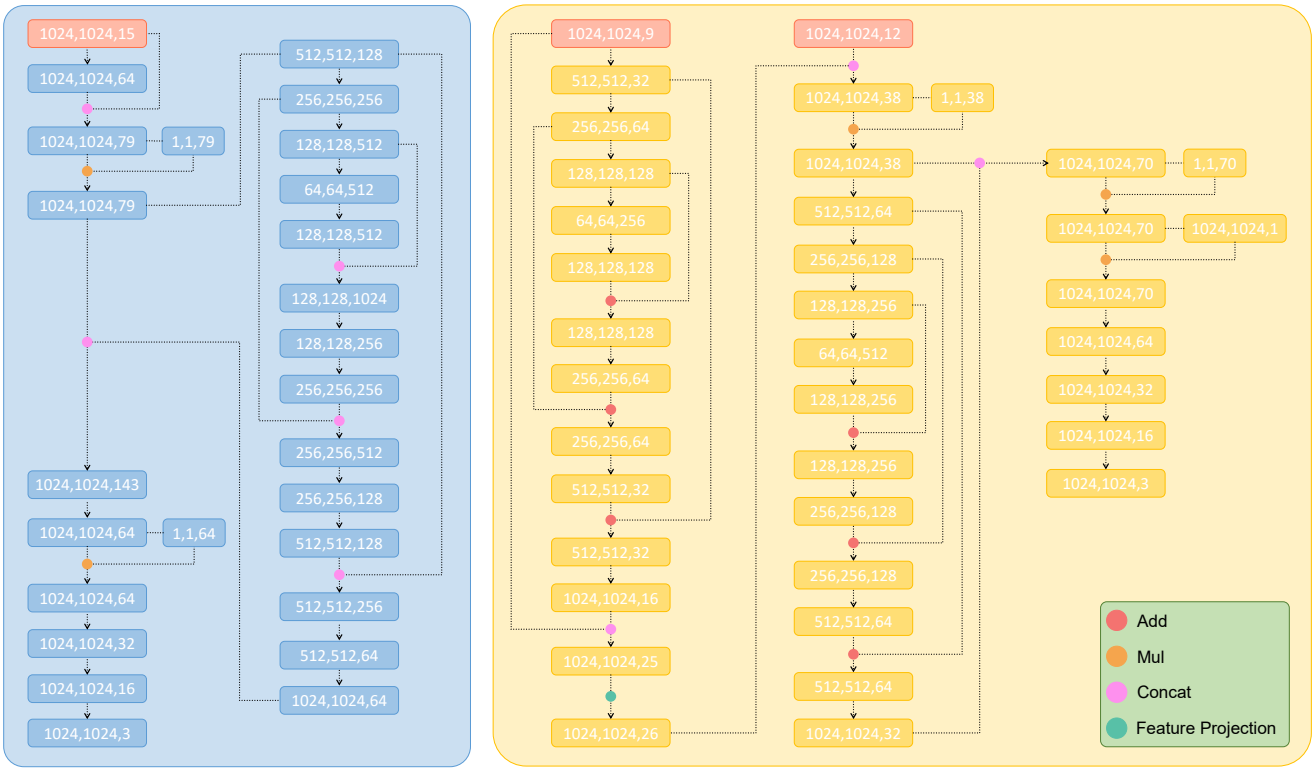

Figure S3. **Network Architecture**. The inputs are: (1) **(1024,1024,15)**, hole mask, holed uv, holed displacement uv, sub-smplx template normal uv, sub-smplx template uv. The sub-smplx template is in star pose. (2) **(1024,1024,9)**, image, front normal map, back normal map. (3) **(1024,1024,12)**, smoothed uv, normal uv, posed sub-smplx uv, posed sub-smplx normal uv.

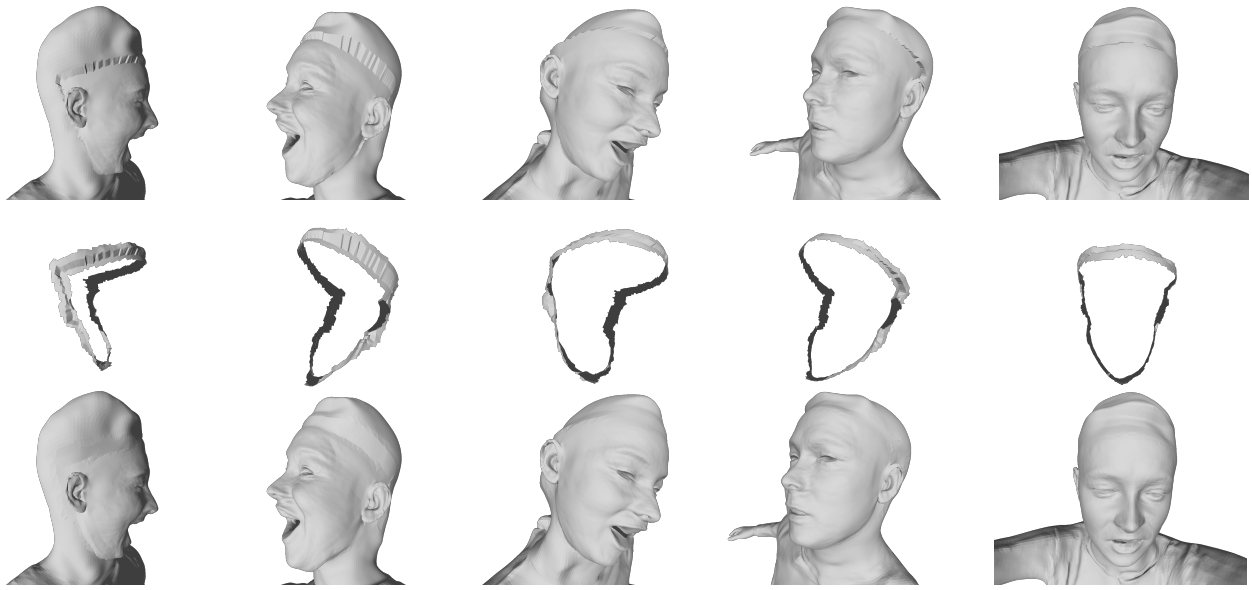

Figure S4. **Face edge smoothing**. The first line shows the results after face substitution, the second line shows the replaced edges, and the third line shows the smoothed results.

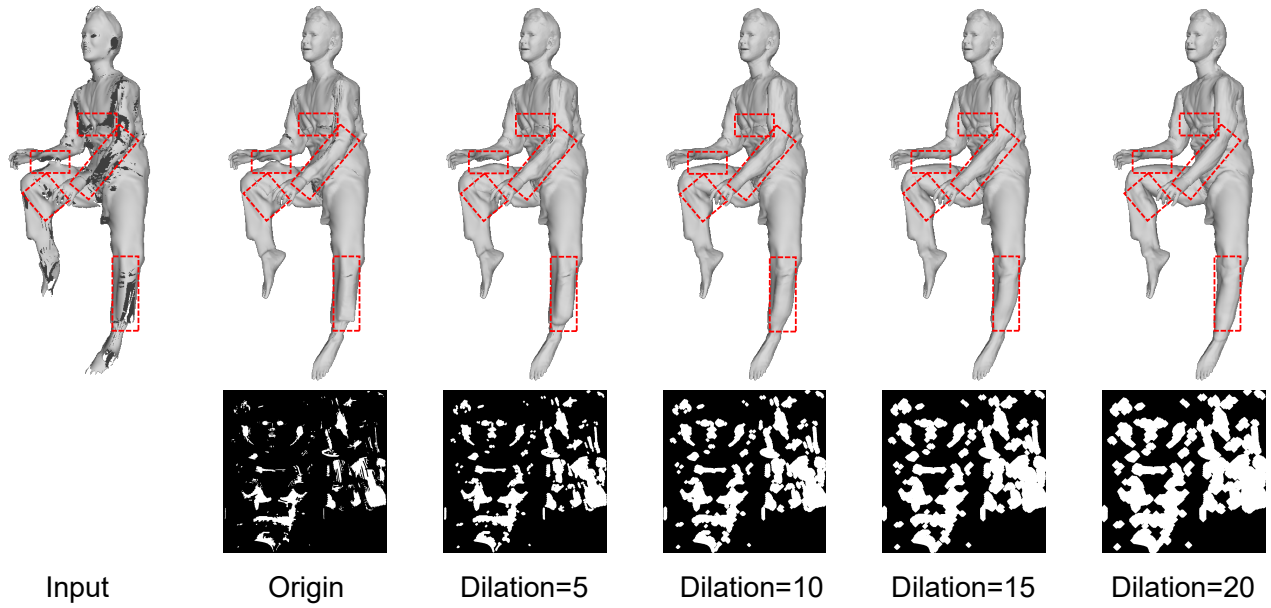

Figure S5. **Mask dilation in completion**. We display the completed results of different hole masks in the first line. The corresponding dilated hole masks are shown below.

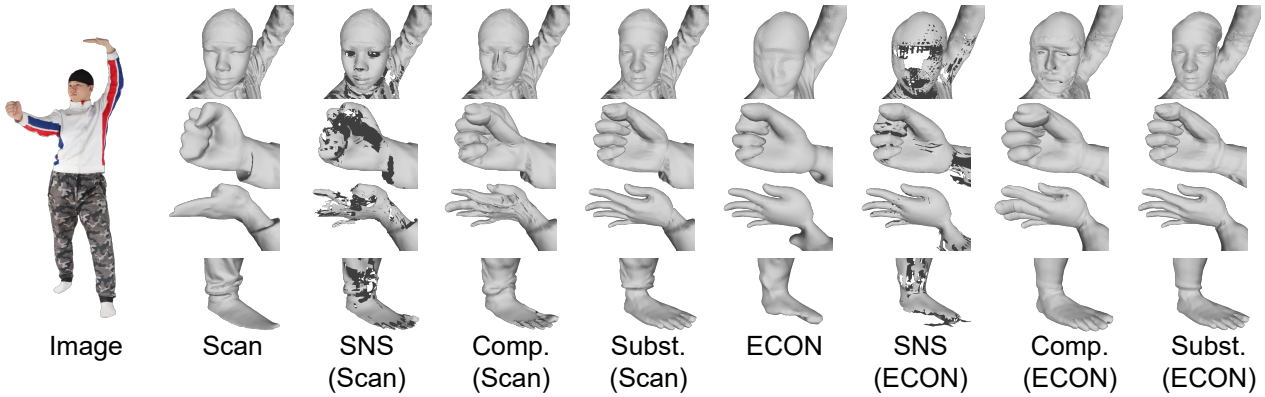

Figure S6. **The completion results of the face, hands, and feet**.

Figure S7. **The refine iteration for textured mesh**.

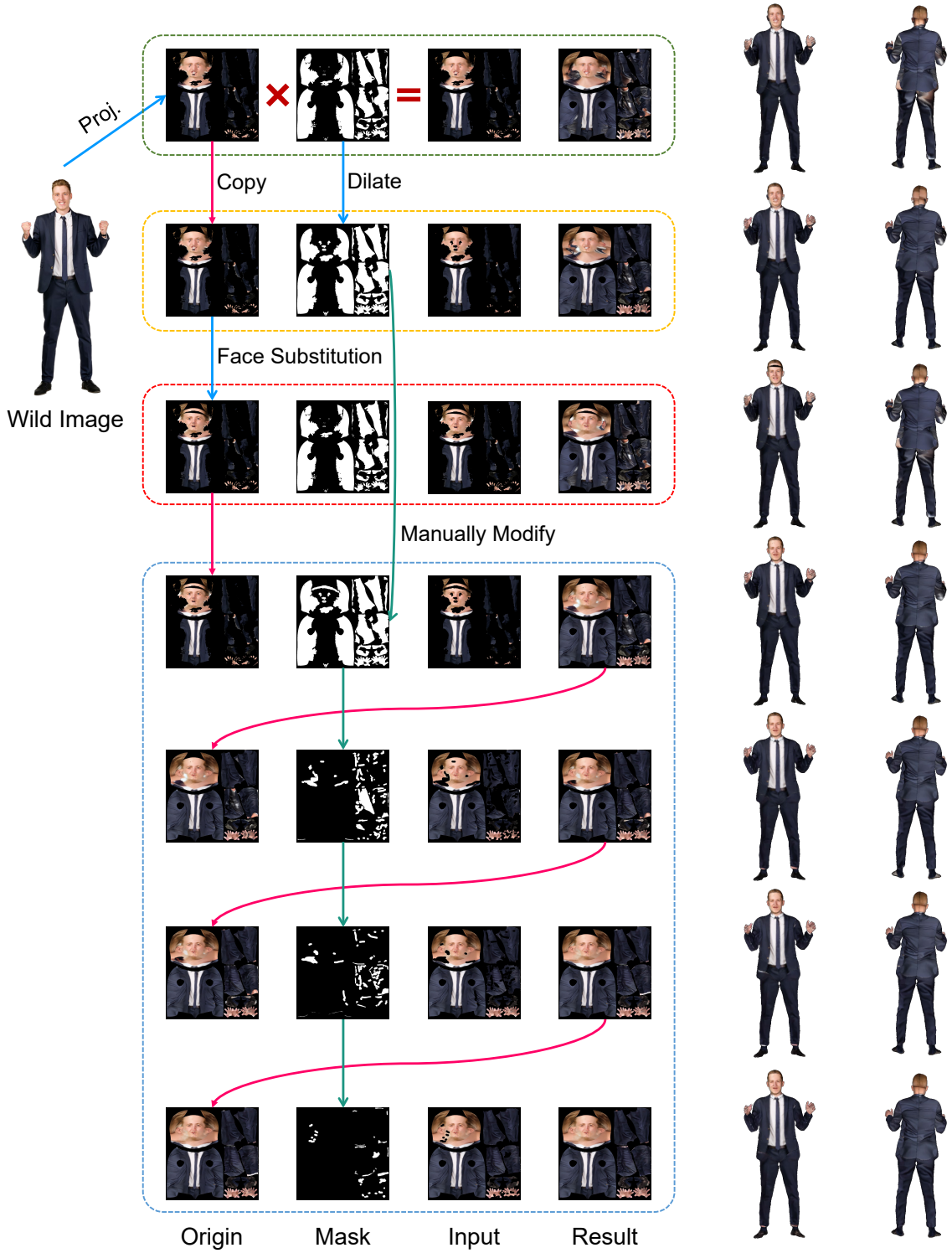

Figure S8. **Texture Inpainting**. We present four different strategies for inpainting the partial texture using SHERT. Green: Using the projected partial texture and mask directly. Yellow: Using the dilated mask. Red: Using the substituted facial texture. Blue: Manual iterative optimization. The corresponding rendering results are displayed on the right.

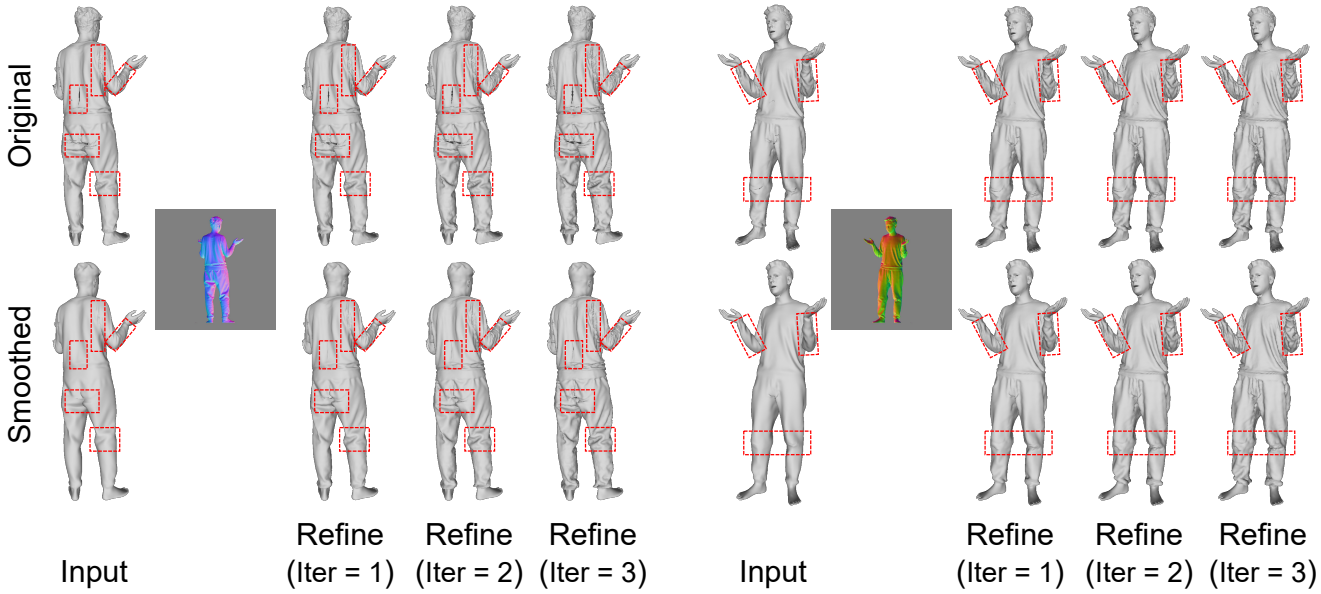

Figure S9. **The influence of smoothed inputs for refinement**. We show the refinement results of the original input in the first line and the smoothed input in the second line.

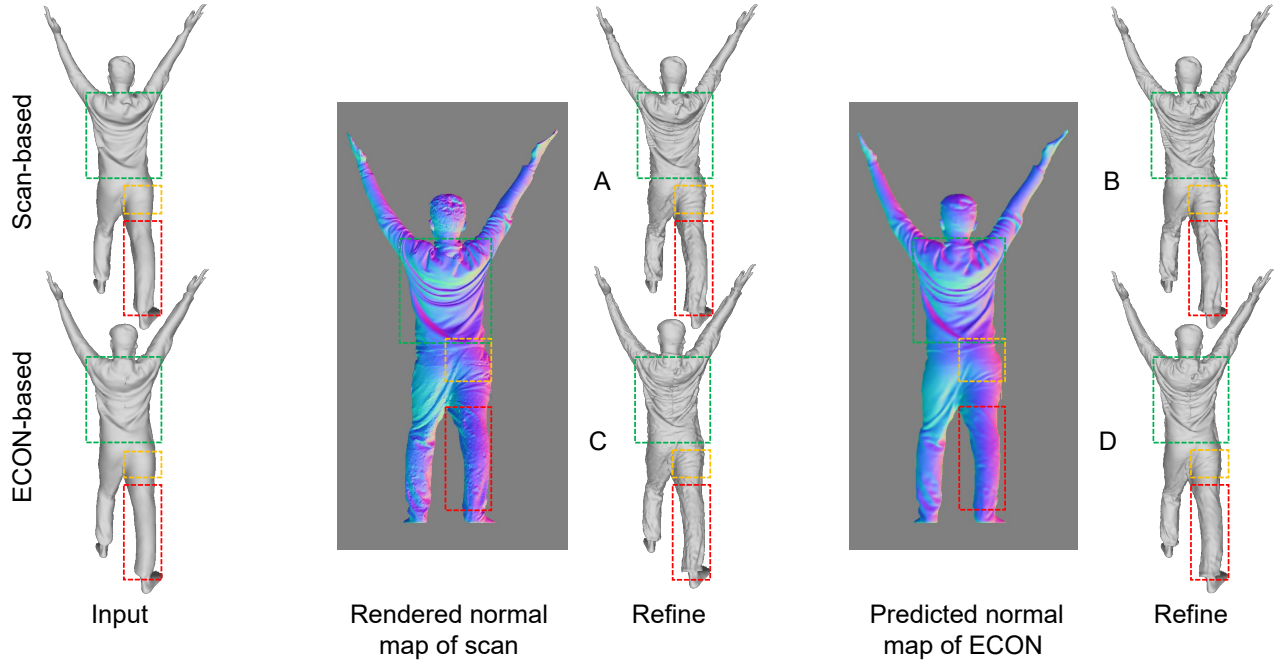

Figure S10. **The effects of normal quality**. We combine the mesh and normal obtained from the THuman2.0 scan, as well as the mesh and normal obtained through econ, to display four different results using the refinement network. The first line (A, B) uses the scan-based completed mesh as input, while the second line (C, D) uses the ECON-based completed mesh. The results shown on the left (A, C) utilize the scan-rendered normal map, while the ones on the right (B, D) utilize the ECON-predicted normal map. Green: When using matched normal maps (A, D), the enhancement of details is consistent with the geometric features of the model, resulting in stacked optimization results. However, when inconsistent inputs are used (B, C), the spatial position of the normal features does not correspond exactly to the geometric features of the mesh itself, resulting in a misalignment between the optimized details and the original geometric details. Yellow: When clear and detailed normal map (A, C) is inputted, the geometric details corresponding to the normal map can be optimized well, regardless of whether the input mesh matches the normal map. Conversely, if the details on the normal map are not clear enough (B, D), the details on the final optimized results will also be relatively blurred. Red: When there are noise in the normal map (A, C), the refinement network mistakenly adds this noise as details to the inputs. Conversely, a clean normal input (B, D) can effectively avoid such situation.

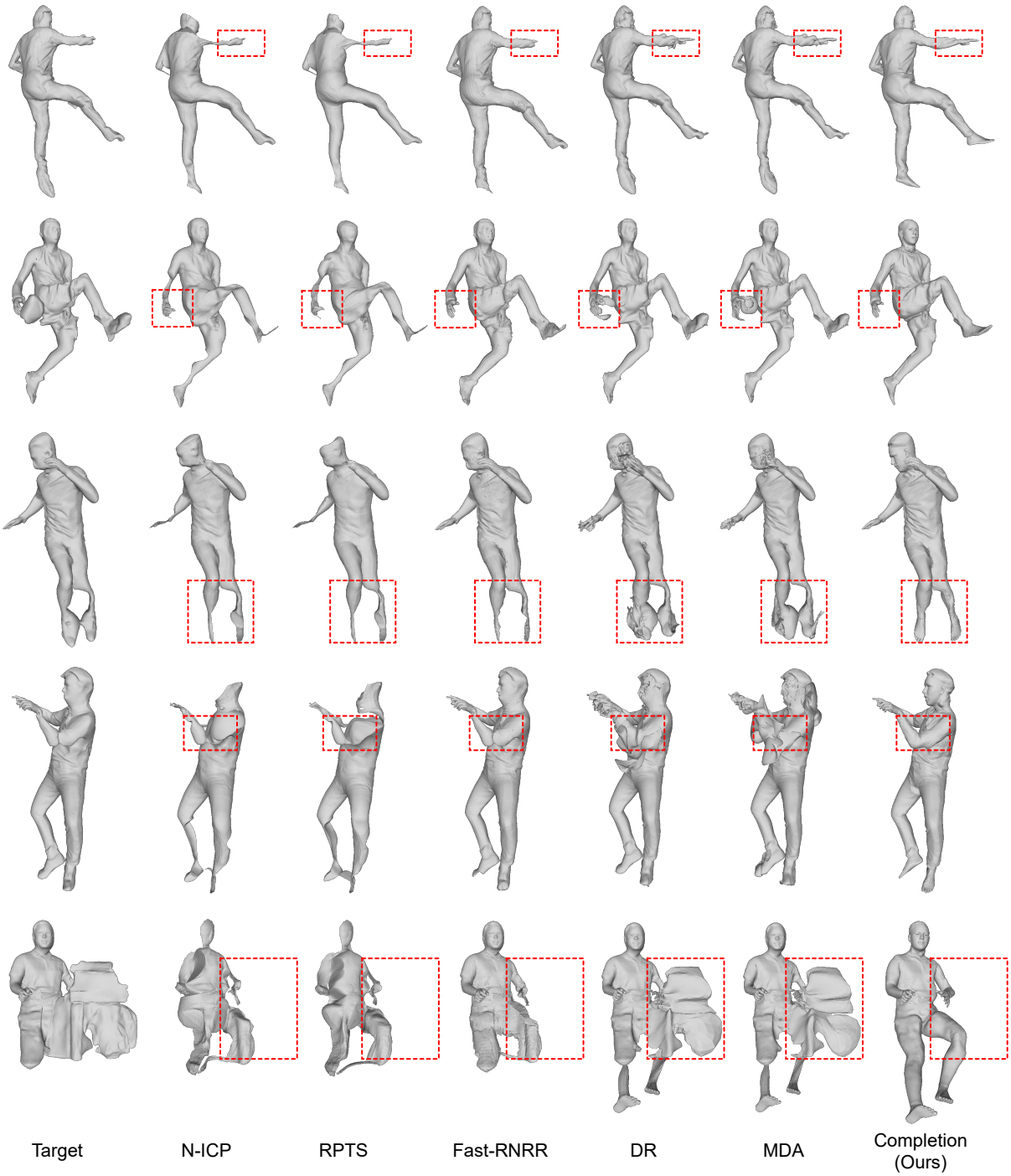

Figure S11. **Registration results for incomplete and incorrect inputs**. The results indicate that the SNS and completion processes of SHERT have significantly better robustness than other registration methods (N-ICP [6], RPTS [36], Fast-RNRR [69], DR [46], MDA [30]).

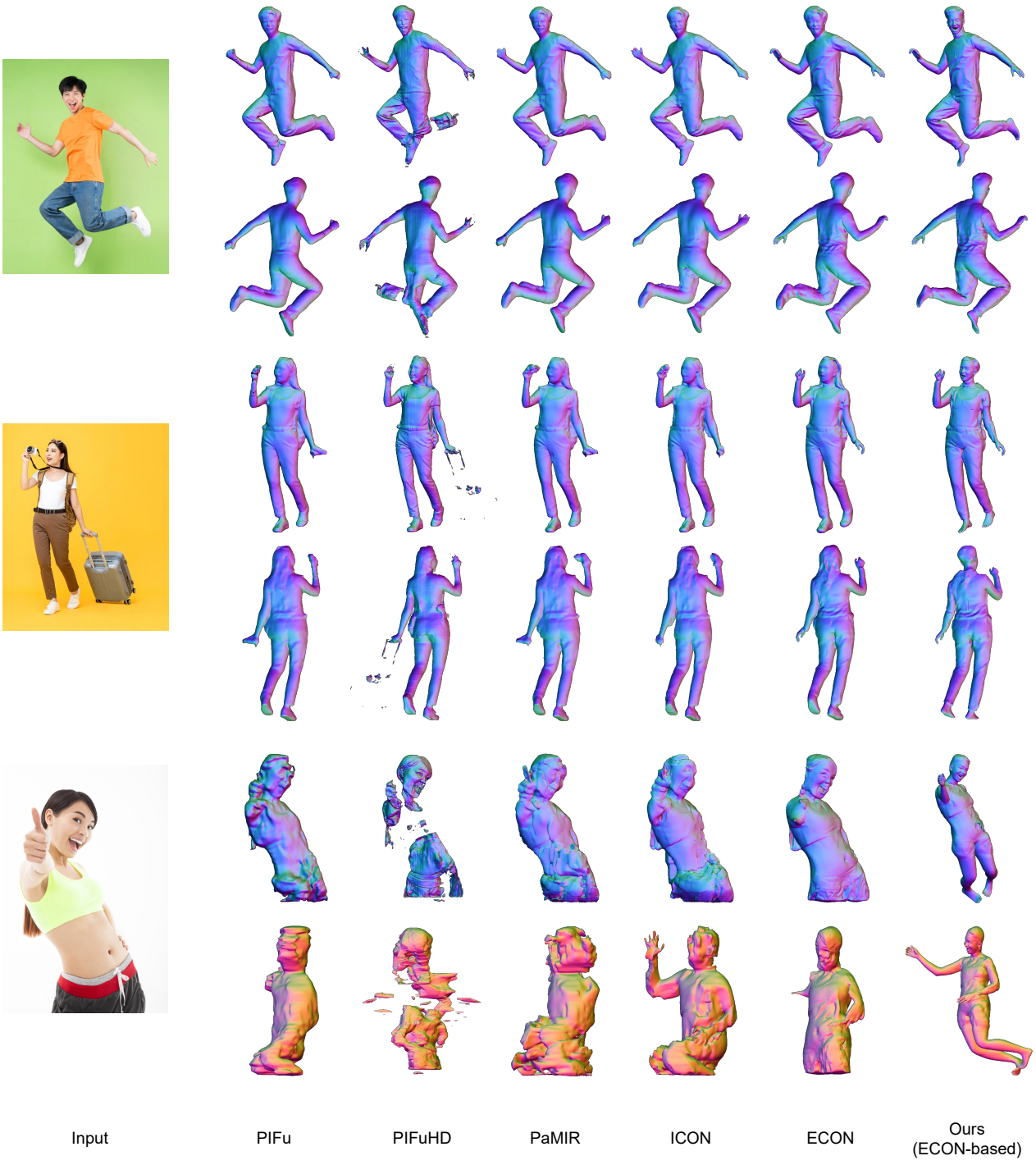

Figure S12. **Comparisons for monocular image reconstruction on in-the-wild images**. We compare the reconstruction quality of various methods including PIFu [55], PIFuHD [56], PaMIR [71], ICON [64], ECON [65], and our SHERT. SHERT takes the ECON's result as the target surface. We mainly showcase the performance of SHERT when facing the complex pose (line 1-2), the long-haired person (line 3-4), and the incomplete input (line 5-6).

Figure S13. **Texture inpainting results on in-the-wild images**.

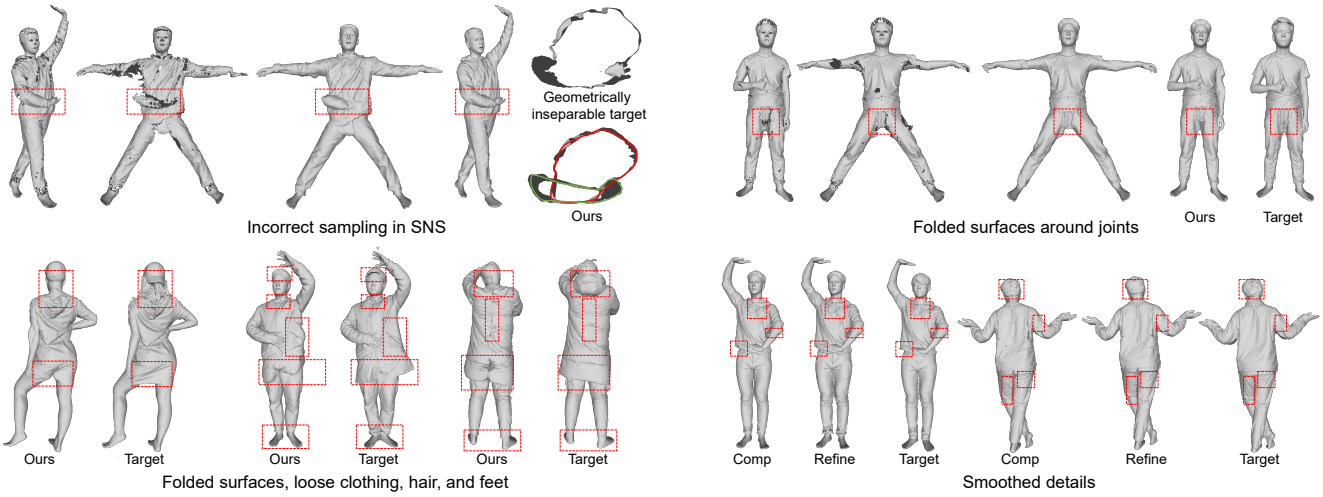

Figure S14. **Limitations**.

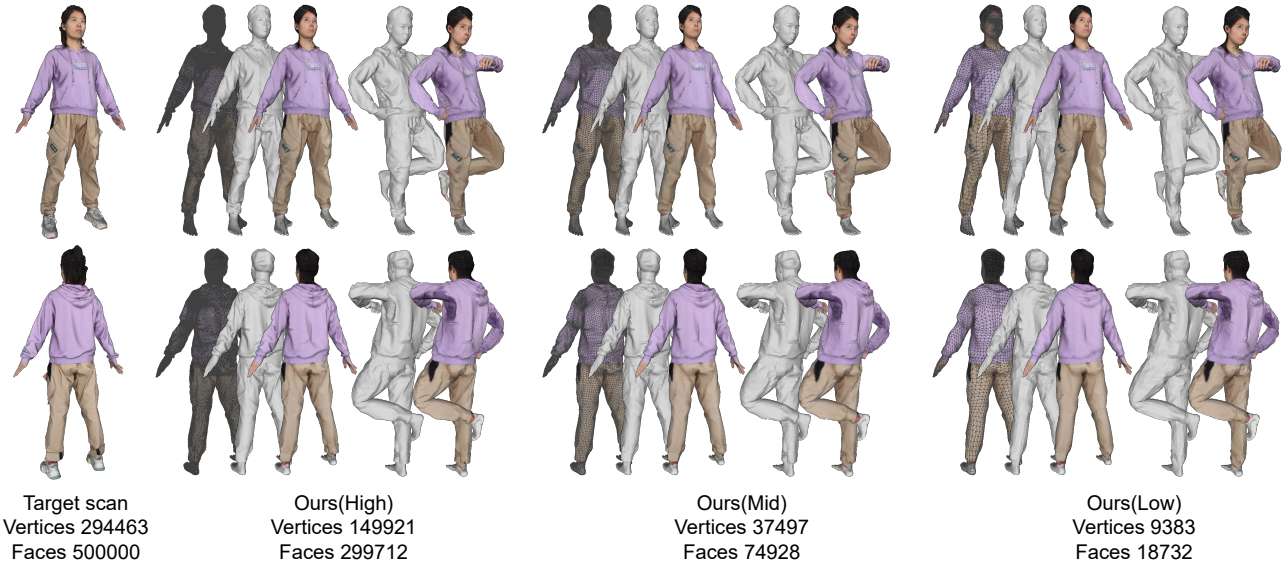

Figure S15. **Mesh down-sampling**. SHERT makes it possible to efficiently generate high-fidelity low-resolution models from a high-resolution model by using pre-defined human model templates.

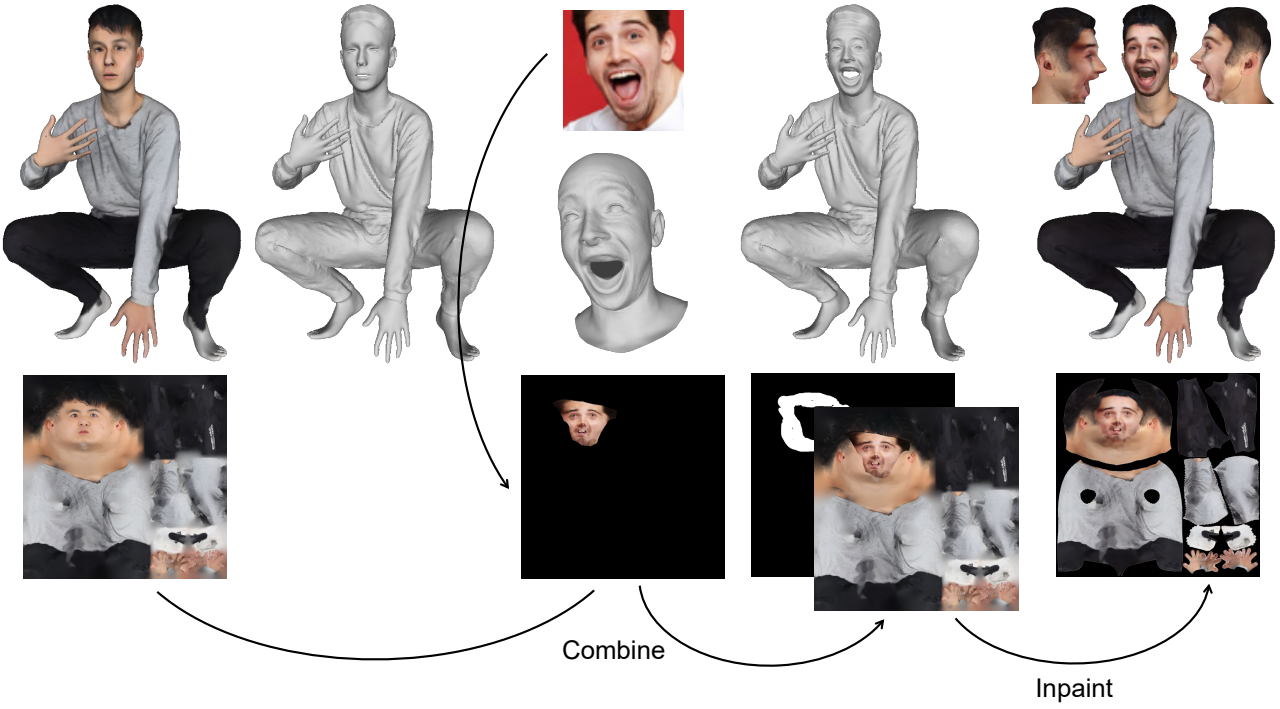

Figure S16. **3D face substitution**.