Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning – Supplemental Materials

Shichen Liu[1,2], Tianye Li[1,2], Weikai Chen[1], and Hao Li[1,2,3]

[1]USC Institute for Creative Technologies
[2]University of Southern California
[3]Pinscreen

{lshichen, tli, wechen}@ict.usc.edu   hao@hao-li.com

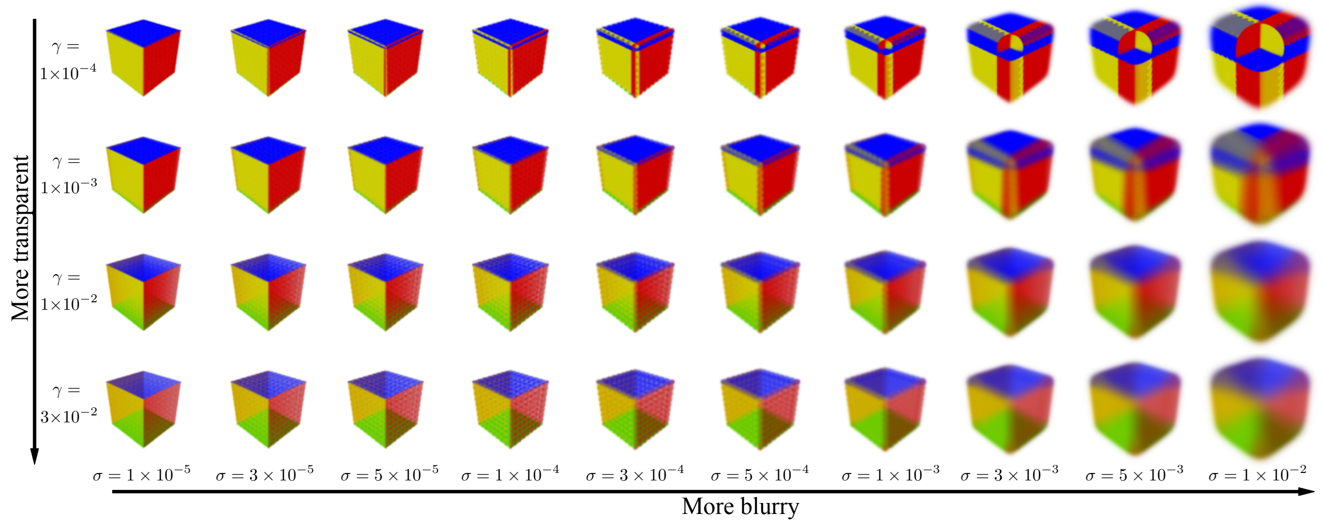
$\gamma = 1\times10^{-4}$
$\gamma = 1\times10^{-3}$
$\gamma = 1\times10^{-2}$
$\gamma = 3\times10^{-2}$
More transparent
$\sigma = 1\times10^{-5}$
$\sigma = 3\times10^{-5}$
$\sigma = 5\times10^{-5}$
$\sigma = 1\times10^{-4}$
$\sigma = 3\times10^{-4}$
$\sigma = 5\times10^{-4}$
$\sigma = 1\times10^{-3}$
$\sigma = 3\times10^{-3}$
$\sigma = 5\times10^{-3}$
$\sigma = 1\times10^{-2}$
More blurry

Figure 1: Different rendering effects achieved by our proposed SoftRas renderer. We show how a colorized cube can be rendered in various ways by tuning the parameters of SoftRas. In particular, by increasing γ, SoftRas can render the object with more tranparency while more blurry renderings can be achieved via increasing σ. As $\gamma \to 0$ and $\sigma \to 0$, one can achieve rendering effect closer to standard rendering.

1. Gradient Computation

In this section, we provide more analysis on the variants of the probability representation (main paper Section 3.2) and aggregate function (main paper Section 3.3), in terms of the mathematical formulation and the resulting impact on the backward gradient.

1.1. Overview

According to the computation graph in Figure 3 of the main paper, our gradient from rendered image $\mathbf{I}$ to vertices in mesh $\mathbf{M}$ is obtained by

$$\frac{\partial \mathbf{I}}{\partial \mathbf{M}} = \frac{\partial \mathbf{I}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \mathbf{M}} + \frac{\partial \mathbf{I}}{\partial \mathbf{Z}}\frac{\partial \mathbf{Z}}{\partial \mathbf{M}} + \frac{\partial \mathbf{I}}{\partial \mathbf{N}}\frac{\partial \mathbf{N}}{\partial \mathbf{M}}. \tag{1}$$

While $\frac{\partial \mathbf{U}}{\partial \mathbf{M}}, \frac{\partial \mathbf{Z}}{\partial \mathbf{M}}, \frac{\partial \mathbf{I}}{\partial \mathbf{N}}$ and $\frac{\partial \mathbf{N}}{\partial \mathbf{M}}$ can be easily obtained by inverting the projection matrix and the illumination models, $\frac{\partial \mathbf{I}}{\partial \mathbf{U}}$ and $\frac{\partial \mathbf{I}}{\partial \mathbf{Z}}$ do not exist in conventional rendering pipelines. Our framework introduces an intermediate representation, probability map $\mathcal{D}$, that factorizes the gradient $\frac{\partial \mathbf{I}}{\partial \mathbf{U}}$ to $\frac{\partial \mathbf{I}}{\partial \mathcal{D}}\frac{\partial \mathcal{D}}{\partial \mathbf{U}}$, enabling the differentiability of $\frac{\partial \mathbf{I}}{\partial \mathbf{U}}$. Further, we obtain $\frac{\partial \mathbf{I}}{\partial \mathbf{Z}}$ via the proposed aggregate function. In the following context, we will first address the gradient $\frac{\partial \mathcal{D}}{\partial \mathbf{U}}$ in Section 1.2 and gradient $\frac{\partial \mathbf{I}}{\partial \mathcal{D}}$ and $\frac{\partial \mathbf{I}}{\partial \mathbf{Z}}$ in Section 1.3.

1.2. Probability Map Computation

The probability maps $\{\mathcal{D}_j^i\}$ based on the relative position between a given triangle f_j and pixel p_i are obtained via *sigmoid function* with temperature σ and distance metric $D(i,j)$:

$$\mathcal{D}_j^i = \frac{1}{1 + \exp\left(-\frac{D(i,j)}{\sigma}\right)}, \tag{2}$$

where the metric D essentially satisfies: (1) $D(i,j) > 0$ if p_i lies inside f_j; (2) $D(i,j) < 0$ if p_i lies outside f_j and (3) $D(i,j) = 0$ if p_i lies exactly on the boundary of f_j. The positive scalar σ controls the sharpness of the probability, where $\mathcal{D}_j$ converges to a binary mask as $\sigma \to 0$.

We introduce two candidate metrics, namely *signed Euclidean distance* and *barycentric metric*. We represent p_i using *barycentric coordinate* $\mathbf{b}_j^i \in \mathbb{R}^3$ defined by f_j:

$$\mathbf{b}_j^i = \mathbf{U}_j^{-1}\mathbf{p}_i, \tag{3}$$

where $\mathbf{U}_j = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}_{f_j}$ and $\mathbf{p}_i = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{p_i}$.

1.2.1 Euclidean Distance

Let $\mathbf{t}_j^i \in \mathbb{R}^3$ be the barycentric coordinate of the point on the edge of f_j that is closest to p_i. The signed Euclidean distance $D_E(i,j)$ from p_i to the edges of f_j can be computed as:

$$\begin{aligned} D_E(i,j) &= \delta_j^i \left\| \mathbf{U}_j(\mathbf{t}_j^i - \mathbf{b}_j^i) \right\|_2^2 \\ &= \delta_j^i \left\| \mathbf{U}_j \mathbf{t}_j^i - \mathbf{p}_i \right\|_2^2, \end{aligned} \tag{4}$$

where δ_j^i is a sign indicator defined as $\delta_j^i = \{+1, \text{if } p_i \in f_j; -1, \text{otherwise}\}$.

Then the partial gradient $\frac{\partial D_E(i,j)}{\partial \mathbf{U}_j}$ can be obtained via:

$$\frac{\partial D_E(i,j)}{\partial \mathbf{U}_j} = 2\delta_j^i \left(\mathbf{U}_j \mathbf{t}_j^i - \mathbf{p}_i\right)\left(\mathbf{t}_j^i\right)^T. \tag{5}$$

1.2.2 Barycentric Metric

We define the barycentric metric $D_B(i,j)$ as the minimum of barycentric coordinate:

$$D_B(i,j) = \min\{\mathbf{b}_j^i\} \tag{6}$$

let $s = \underset{k}{\operatorname{argmin}}\,(\mathbf{b}_j^i)^{(k)}$, then the gradient from $D_B(i,j)$ to $\mathbf{U}_j$ can be obtained through:

$$\begin{aligned} \frac{\partial D_B(i,j)}{\partial \left(\mathbf{U}_j\right)^{(k,l)}} &= \frac{\partial \min\{\mathbf{b}_j^i\}}{\partial \left(\mathbf{U}_j\right)^{(k,l)}} \\ &= \frac{\partial \left(\mathbf{b}_j^i\right)^{(s)}}{\partial \mathbf{U}_j^{-1}} \frac{\partial \mathbf{U}_j^{-1}}{\partial \left(\mathbf{U}_j\right)^{(k,l)}} \\ &= -\sum_t \left(\mathbf{p}_i\right)^{(t)} \left(\mathbf{U}_j^{-1}\right)^{(s,k)} \left(\mathbf{U}_j^{-1}\right)^{(l,t)}, \end{aligned} \tag{7}$$

where k and l are the indices of $\mathbf{U}_j$'s element.

1.3. Aggregate function

1.3.1 Softmax-based Aggregate Function

According to $\mathcal{A}_S(\cdot)$, the output color is:

$$I^i = \mathcal{A}_S(\{C_j^i\}) = \sum_j w_j^i C_j^i + w_b^i C_b, \tag{8}$$

where the weight $\{w_j\}$ is obtained based on the relative depth $\{z_j\}$ and the screen-space position of triangle f_j and pixel p_i as indicated in the following equation:

$$w_j^i = \frac{\mathcal{D}_j^i \exp\left(z_j^i/\gamma\right)}{\sum_k \mathcal{D}_k^i \exp\left(z_k^i/\gamma\right) + \exp\left(\epsilon/\gamma\right)}; \tag{9}$$

C_b and w_b^i denote the color and weight of background respectively where

$$w_b^i = \frac{\exp\left(\epsilon/\gamma\right)}{\sum_k \mathcal{D}_k^i \exp\left(z_k^i/\gamma\right) + \exp\left(\epsilon/\gamma\right)}; \tag{10}$$

z_j^i is the clipped normalized depth. Note that we normalize the depth so that the closer triangle receives a larger z_j^i by

$$z_j^i = \frac{Z_{far} - Z_j^i}{Z_{far} - Z_{near}}, \tag{11}$$

where Z_j^i denotes the actual clipped depth of f_j at p_i, while Z_{near} and Z_{far} denote the far and near cut-off distances of the viewing frustum.

Specifically, the aggregate function $\mathcal{A}_S(\cdot)$ satisfies the following three properties: (1) as $\gamma \to 0$ and $\sigma \to 0$, w^i converges to an one-hot vector where only the closest triangle contains the projection of p_i is one, which shows the consistency between $\mathcal{A}_S(\cdot)$ and *z-buffering*; (2) w_b^i is close to one only when there is no triangle that covers p_i; (3) $\{w_j^i\}$ is robust to z-axis translation. In addition, γ is a positive scalar that could balance out the scale change on z-axis.

The gradient $\frac{\partial I}{\partial \mathcal{D}_j^i}$ and $\frac{\partial I}{\partial z_j^i}$ can be obtained as follows:

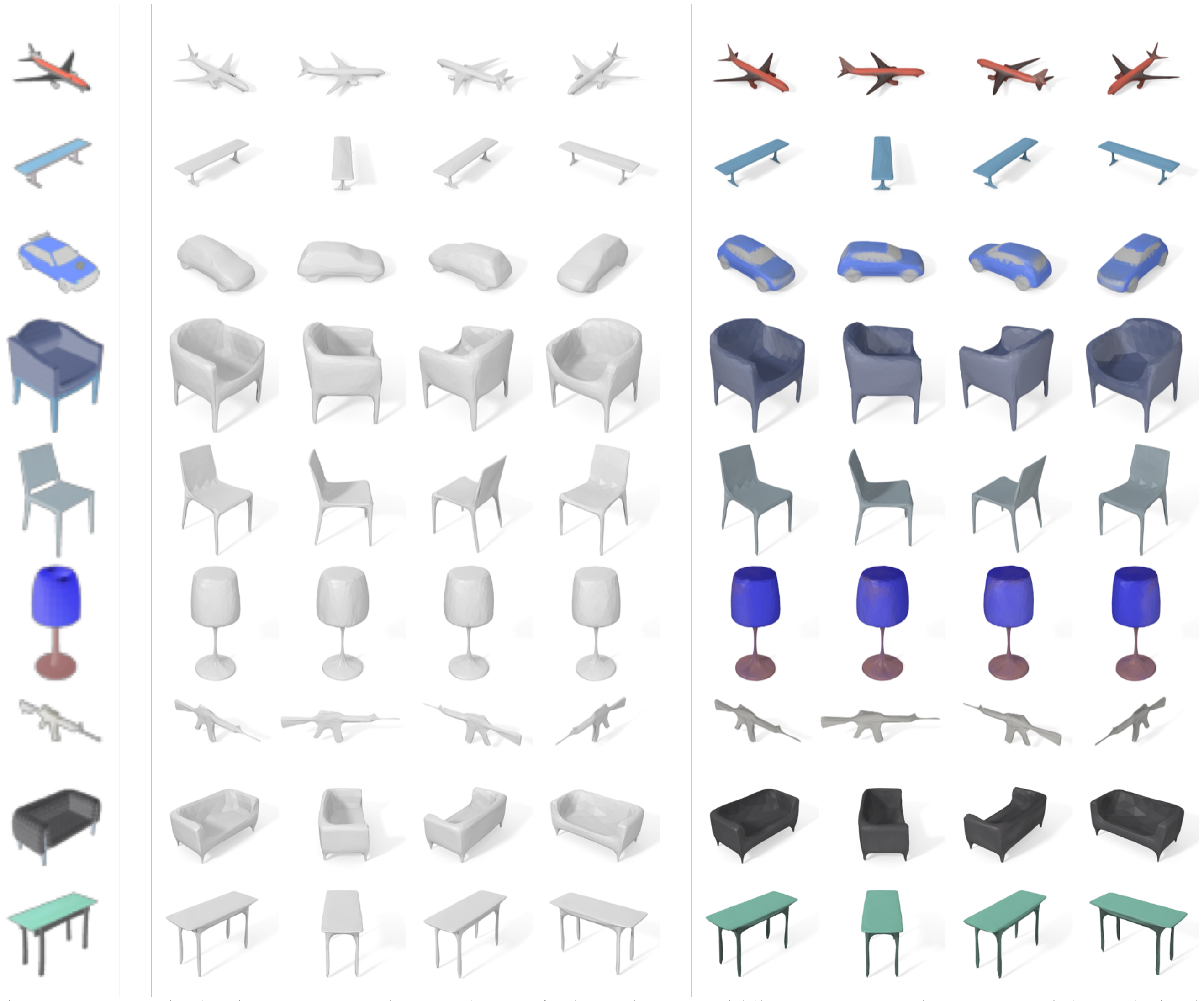
Figure 2: More single-view reconstruction results. Left: input image; middle: reconstructed geometry; right: colorized reconstruction.

$$
\begin{aligned}
\frac{\partial I^i}{\partial \mathcal{D}_j^i} &= \sum_k \frac{\partial I^i}{\partial w_k^i} \frac{\partial w_k^i}{\partial \mathcal{D}_j^i} + \frac{\partial I^i}{\partial w_b^i} \frac{\partial w_b^i}{\partial \mathcal{D}_j^i} \\
&= \sum_{k \neq j} -C_k^i \frac{w_j^i w_k^i}{\mathcal{D}_j^i} + C_j^i \left(\frac{w_j^i}{\mathcal{D}_j^i} - \frac{w_j^i w_j^i}{\mathcal{D}_j^i}\right) - C_b^i \frac{w_j^i w_b^i}{\mathcal{D}_j^i} \\
&= \frac{w_j^i}{\mathcal{D}_j^i}(C_j^i - I^i) \qquad (12)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial I^i}{\partial z_j^i} &= \sum_k \frac{\partial I^i}{\partial w_k^i} \frac{\partial w_k^i}{\partial z_j^i} + \frac{\partial I^i}{\partial w_b^i} \frac{\partial w_b^i}{\partial z_j^i} \\
&= \sum_{k \neq j} -C_k^i \frac{w_j^i w_k^i}{\gamma} + C_j^i \left(\frac{w_j^i}{\gamma} - \frac{w_j^i w_j^i}{\gamma}\right) - C_b^i \frac{w_j^i w_b^i}{\gamma} \\
&= \frac{w_j^i}{\gamma}(C_j^i - I^i) \qquad (13)
\end{aligned}
$$

1.3.2 Occupancy Aggregate Function

Independent from color and illumination, the silhouette of the object can be simply described by an occupancy aggregate function $\mathcal{A}_O(\cdot)$ as follows:

$$
I_{sil}^i = \mathcal{A}_O(\{\mathcal{D}_j^i\}) = 1 - \prod_j (1 - \mathcal{D}_j^i). \qquad (14)
$$

Hence, the partial gradient $\frac{\partial I_{sil}^i}{\partial \mathcal{D}_j^i}$ can be computed as follows:

$$
\frac{\partial I_{sil}^i}{\partial \mathcal{D}_j^i} = \frac{1 - I_{sil}^i}{1 - \mathcal{D}_j^i}. \qquad (15)
$$

2. Forward Rendering Results

As demonstrated in Figure 1, our framework is able to directly render a given mesh, which cannot be achieved by any existing rasterization-based differentiable renderers [1, 2]. In addition, compared to standard graphics renderer, SoftRas can achieve different rendering effects in a continuous manner thanks to its probabilistic formulation. Specifically, by increasing σ, the key parameter that controls the sharpness of the screen-space probability distribution, we are able to generate more blurry rendering results. Furthermore, with increased γ, one can assign more weights to the triangles on the far end, naturally achieving more transparency in the rendered image. As discussed in Section 5.2 of the main paper, the blurring and transparent effects are the key for reshaping the energy landscape in order to avoid local minima.

3. Network Structure

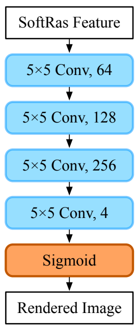

Figure 3: Network Architecture of $\mathcal{A}_N$, an alternative color aggregate function that is implemented as a neural networks.

We provide detailed structures for all neural networks that were mentioned in the main paper. Figure 3 shows the structure of $\mathcal{A}_N$ (Section 3.3 and 5.1.4 of the main paper), an alternative color aggregate function that is implemented as a neural network. In particular, input SoftRas features are first passed to four consecutive convolutional layers and then fed into a sigmoid layer to model non-linearity. We train $\mathcal{A}_N$ with the output of a standard rendering pipeline as ground truth to achieve a *parametric* differentiable renderer.

We employ an encoder-decoder architecture for our single-view mesh reconstruction. The encoder is used as a feature extractor, whose network structure is shown in Figure 4. The detailed network structure of the color and shape generators are illustrated in Figure 5(a) and (b) respectively. Both networks (Figure 6 of the main paper) share the same feature extractor. The shape generators consists of three fully connected layers and outputs a per-vertex displacement vector that deforms a template mesh into a target model. The color generator contains two fully connected streams: one for sampling the input image to build the color palette and the other one for selecting colors from the color palette to texture the sampling points.

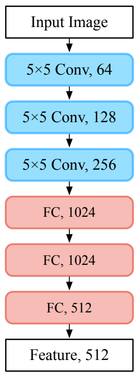

Figure 4: Network architecture of the feature extractor.

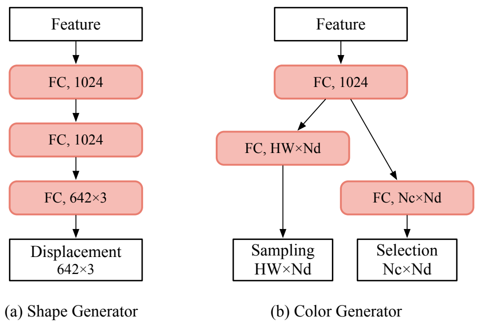

Figure 5: Network architectures of the shape and color generator.

4. More Results on Image-based 3D Reasoning

We show more results on single-view mesh reconstruction and image-base shape fitting.

4.1. Single-view Mesh Reconstruction

4.1.1 Intermediate Mesh Deformation

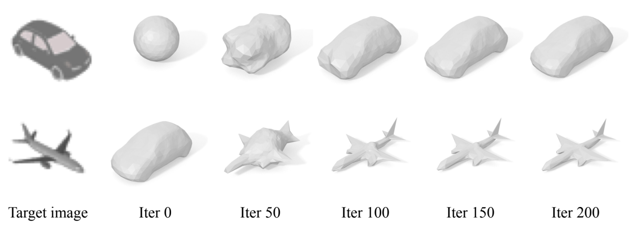

Figure 6: Visualization of intermediate mesh deformation during training. First row: the network deforms the input sphereto a desired car model that corresponds to the target image. Second row: the generated car model is further deformed to reconstruct the airplane.

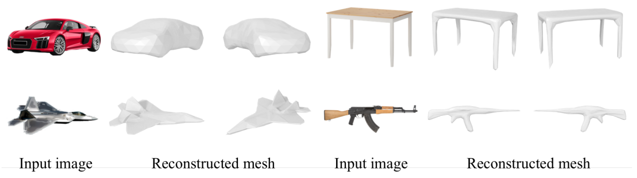

Figure 7: Single-view reconstruction results on real images.

In Figure 6, we visualize the intermediate process of how an input mesh is deformed to a target shape after the supervision provided by SoftRas. As shown in the first row, the mesh generator gradually deforms a sphere template to a desired car shape which matches the input image. We then change the target image to an airplane (Figure 6 second row). The network further deforms the generated car model to faithfully reconstruct the airplane. In both examples, the mesh deformation can quickly converge to a high-fidelity reconstruction within 200 iterations, demonstrating the effectiveness of our SoftRas renderer.

4.1.2 Single-view Reconstruction from Real Images

We further evaluate our approach on real images. As demonstrated in Figure 7, though only trained on synthetic data, our model generalizes well to real images and novel views with faithful reconstructions and fine-scale details, e.g. the tail fins of the fighter aircraft and thin structures in the rifle and table legs.

4.1.3 More Reconstruction Results from ShapeNet

We provide more reconstruction results in Figure 2. For each input image, we show its reconstructed geometry (middle) as well as the colored reconstruction (right).

4.2. Fitting Process for Rigid Pose Estimation

We demonstrate the intermediate process of how the proposed SoftRas renderer managed to fit the color cube to the target image in Figure 8. Since the cube is largely occluded, directly leveraging a standard rendering is likely to lead to local minima (Figure 10 of the main paper) that causes non-trivial challenges for any gradient-based optimizer. By rendering the cube with stronger blurring at the earlier stage, our approach is able to avoid local minima, and gradually reduce the rendering loss until an accurate pose can be fitted.

4.3. Visualization of Non-rigid Body Fitting

In Figure 9, we compare the intermediate processes of NMR [1] and SoftRas during the task of fitting the SMPL model to the target pose. As the right hand of subject is completely occluded in the initial image, NMR fails to complete the task due to its incapability of flowing gradient to the occluded vertices. In contrast, our approach is able to obtain the correct pose within 320 iterations thanks to the occlusion-aware technique.

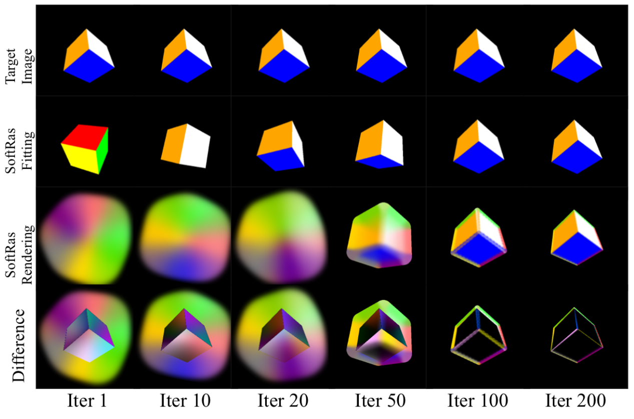

Figure 8: Intermediate process of fitting a color cube (second row) to a target pose shown in the input image (first row). The smoothened rendering (third row) that is used to escape local minimum, as well as the colorized fitting errors (fourth row), are also demonstrated.

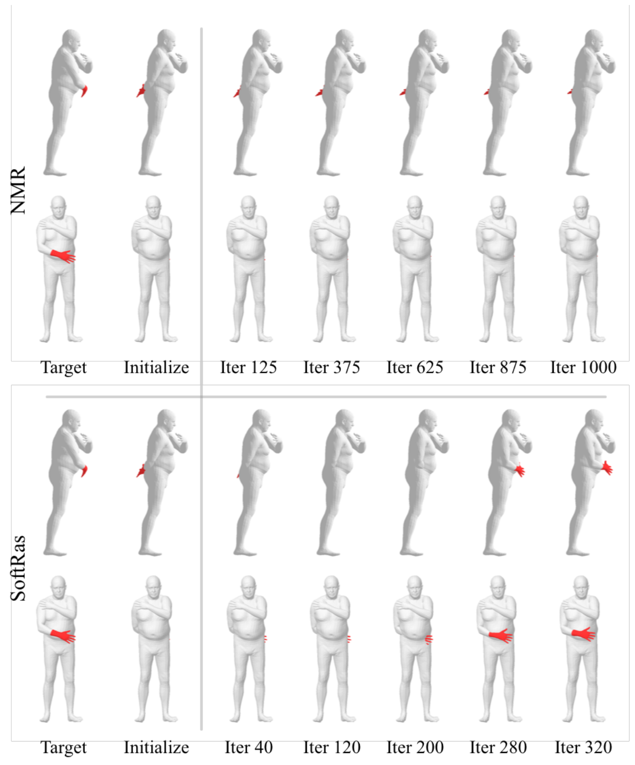

Figure 9: Comparisons of body shape fitting using NMR [1] and our approach. Intermediate fitting processes of both methods are visualized.

References

[1] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. 4, 5

[2] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014. 4