Shape and Viewpoint without Keypoints

Shubham Goel, Angjoo Kanazawa, and Jitendra Malik

UC Berkeley

Supplementary Material

Overview

In this supplementary material, we present additional results including visualization of the shape, texture and camera-multiplex, comparisons to CMR [1], qualitative results on random test samples, an ablation study on texture prediction model, and additional details on the network architecture.

1 More Results

PCA in Texture space In Figure 1, we visualize the learnt texture space by running PCA on predicted uv-textures across the entire test dataset. On the left, the mean texture is a rather dull gray colour, as expected. On the right, we visualize axes of variation. In the first column, we see low-frequency variations in overall colour, head and belly of the bird. In the second column, we see slightly higher-frequency variations that assign different colors to different parts (head, back, belly and wings) of the bird. We can even recognize eyes and beak in the last two rows on the right.

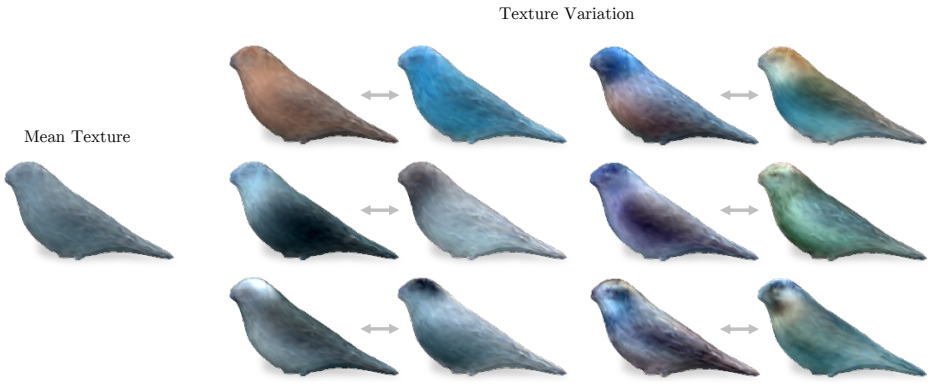

Fig. 1: **PCA in UV-Texture space.** We visualize the learnt texture space by running PCA over all predicted uv-texture maps on the test dataset, and rendering axes of variation on the learnt mean bird shape. See text for discussion.

Learnt shape space for other categories In Figure 2, we compare the input template mesh to the learnt shape space for car, motorbike and shoe categories. Observe, on the left, that the input template and learnt mean shape differ substantially for each of these categories - some more than others. For example, the front tire of the motorbike becomes more significantly more prominent, the back of the car becomes more rounded and the shoe becomes slimmer and more elongated. On the right, we visualize the space of learned shapes by running PCA on all shapes obtained on training and test dataset. The three PCA axis visualized show interesting deformations. For example, in the motorcycle, the three visualized axes vary in prominence of front tire, size of fuel-tank and concavity of seat respectively.

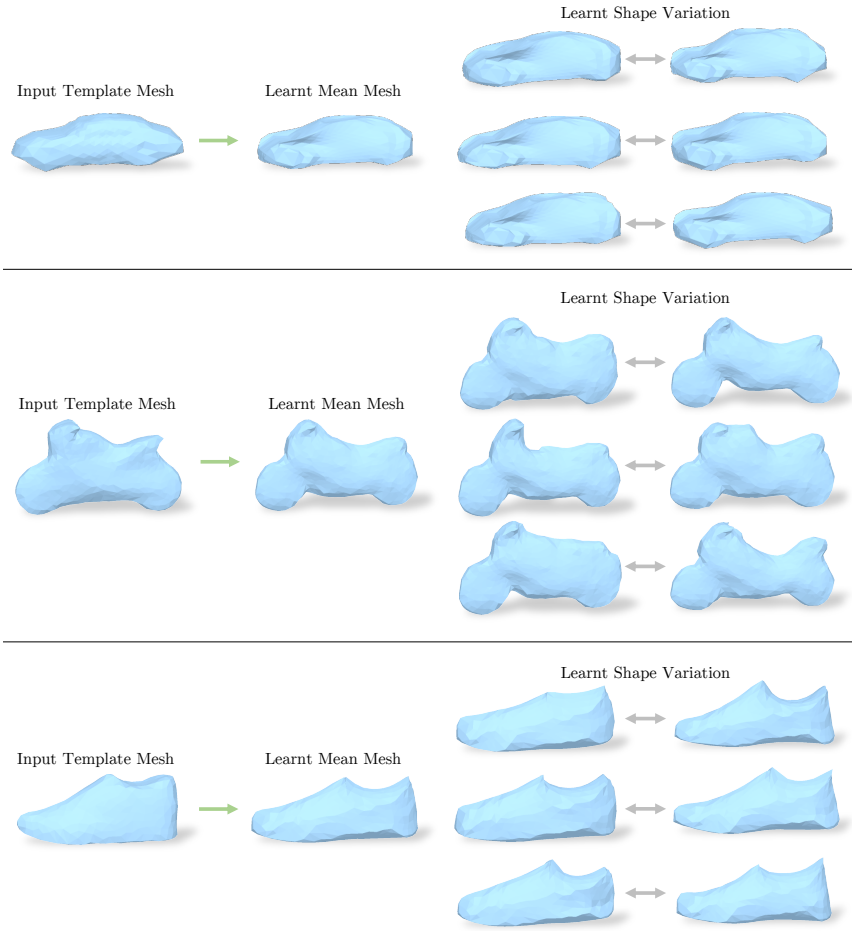

Fig. 2: **Learned Shape on other categories.** On the left, we compare the template shape to the final learnt mean mesh. On the right, we visualize the space of learned shapes by running PCA on all shapes and show three axis of deformations.

Camera-multiplex visualization over time. Figure 3 shows how the azimuth-elevation distribution of camera poses in the camera-multiplex (over the entire training dataset) changes as training progresses. Observe that the distribution changes rapidly initially and results in a final distribution that is very different from the initial distribution.

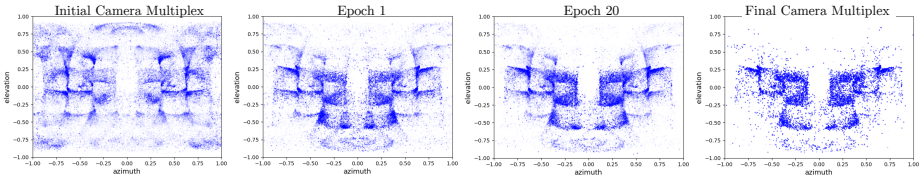

Fig. 3: **Change in Camera Pose Distributions during training.** We show azimuth-elevation scatter plots of $K = 40$ camera poses in all the camera-multiplex of the CUB train set as training progresses. Points corresponding to less probable cameras have a lower alpha value and are more transparent. Starting from top left, we have the camera poses after the camera-multiplex initialization, after 1 training epoch, after 20 training epochs and the final optimized camera poses. The number of camera poses in each multiplex (K) is pruned down from 40 to 4 after epoch 20.

Qualitative comparison to CMR. We qualitatively compare results from U-CMR to 2 variants of CMR [1]. The first is the official CMR model (CMR-official) that was trained using additional keypoint losses and used NMR [2] as it's differentiable renderer. The second is our implementation of CMR (CMR-ours) which is similar to U-CMR in it's architecture for shape/texture, using Softras [3] for rendering, regularizing shape using the graph-laplacian and having the same template mesh as it's initial mean shape, but different from U-CMR in that it uses the ground-truth camera pose from SFM during training. Unlike CMR-official, CMR-ours does not include vertex-keypoint reprojection loss.

In Figures 5-6, we compare CMR-official, CMR-ours and U-CMR. For each input image, the first row is from CMR-official, second is from CMR-ours and the last row is U-CMR. Observe that CMR-official is not as accurate as CMR-ours in capturing the shape and texture of the underlying bird but has pointier beaks and feet because of the keypoint reprojection loss it uses. The figures show that U-CMR shapes are qualitatively very similar to CMR-ours, hence exemplifying our assertion that U-CMR's camera-multiplex optimization alleviates the need for ground-truth camera pose supervision for most cases.

We compare U-CMR and CMR-ours on a random subset of 15 images from the test dataset in Figures 7-8. Observe that U-CMR (second row) accurately predicts shapes that are very similar to those from CMR-ours when the bird is not articulating too much.

Ablation on texture prediction model. We experiment with two different architectures for predicting the texture. First, we explore predicting texture as texture-flow, which is used in CMR. Texture-flow is a 2D positional offset for every pixel in the UV image that specifies where to sample the RGB values from the input image. Second, we predict the UV image values directly (Texture-gen) using a decoder attached to a bottleneck with spatial dimensions preserved. This is the final approach used in U-CMR. We observed that predicting a flow-field can lead to flat degenerate shapes and a collapse of optimized camera poses. Figure 4 shows the collapse in the final optimized camera poses when predicting texture as a flow-field. This is because even when the camera pose is wrong, texture-flow is able to learn to adjust to the bad camera, as the output of the texture-flow across different instances is not necessarily correlated. However, when predicting the texture directly through a decoder, the network learns an implicit spatial prior of the texture across the dataset. For example, the network needs to learn to generate the texture of the eye at the same location in every texture map. Similarly for wings, breast, head etc; as the texture map is predicted in a canonical semantic space. This spatial prior that is learned through a spatial decoder allows the texture prediction to disambiguate incorrect and correct poses in the camera-multiplex.

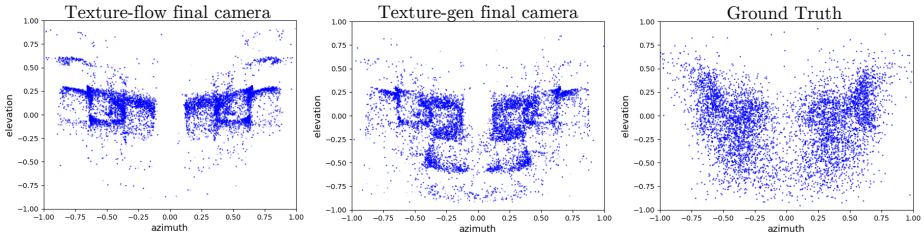

Fig. 4: **Texture-flow camera-multiplex distribution on training set.** On the left, we show the azimuth-elevation distribution of the final camera-multiplex when we predict texture as a flow-field for sampling from the input image. Note how the camera poses have collapsed to 2 broad areas. Center: U-CMR camera-multiplex, Right: GT camera distribution

2 Training details

2.1 Architecture details.

Our code is available on our project page: https://shubham-goel.github.io/ucmr. The shape and texture predictor f has an encoder-decoder architecture. Image $I \in \mathbb{R}^{256\times256\times3}$ is first encoded to latent feature map $z \in \mathbb{R}^{4\times4\times256}$ using Resnet-18. The shape head takes flattened $z \in \mathbb{R}^{16\times256}$ as input and passes it through 2 fully connected layers, each with 200 output channels and then a final linear layer for predicting

$\Delta_V \in \mathbb{R}^{|V|\times 3}$. The texture head takes the latent feature map and bilinearly samples it to $z \in \mathbb{R}^{4\times 8\times 256}$, this is followed by 7 Resnet blocks with 256, 256, 256, 128, 64, 32, 16 output channels respectively, with intermediate bilinear upsampling by a factor of 2 after blocks 1, 3, 4, 5, 6. This is then sent to a final convolution layer that outputs the texture map $I^{uv} \in \mathbb{R}^{128\times 256\times 3}$. The Resnet encoder uses ReLU activations while the shape and texture heads use Leaky-ReLU activations. All networks use batchnorm for normalization. We will release our code upon publication.

After training the shape and texture prediction with camera-multiplex, we learn the feed-forward camera pose predictor g. We attach this as another head to the latent variable z from the shared Resnet-18 trained for shape and texture prediction with camera-multiplex. We freeze the encoder, and then train a fully connected head for predicting camera scale $s \in \mathbb{R}$, translation $t \in \mathbb{R}^2$ and rotation (as quaternion $q \in \mathbb{R}^4$) through 2 fully connected layer each with 200 channels.

Fig. 5: **CMR-official vs CMR-ours vs U-CMR.** We compare U-CMR (third row) to CMR-ours (second row) and CMR-official (first row) on selected images from the test dataset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.

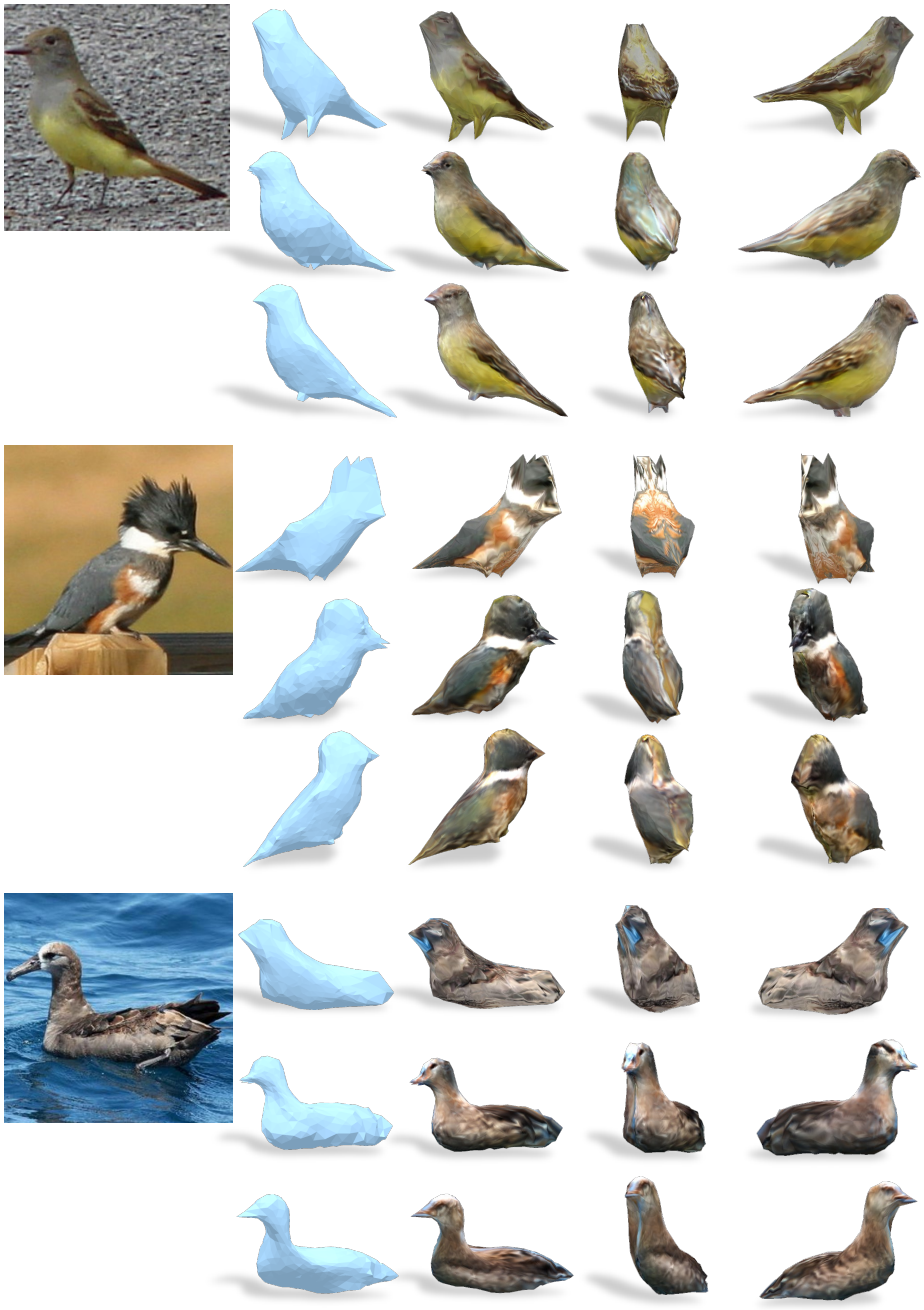

Fig. 6: **CMR-official vs CMR-ours vs U-CMR.** We compare U-CMR (third row) to CMR-ours (second row) and CMR-official (first row) on selected images from the test dataset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.

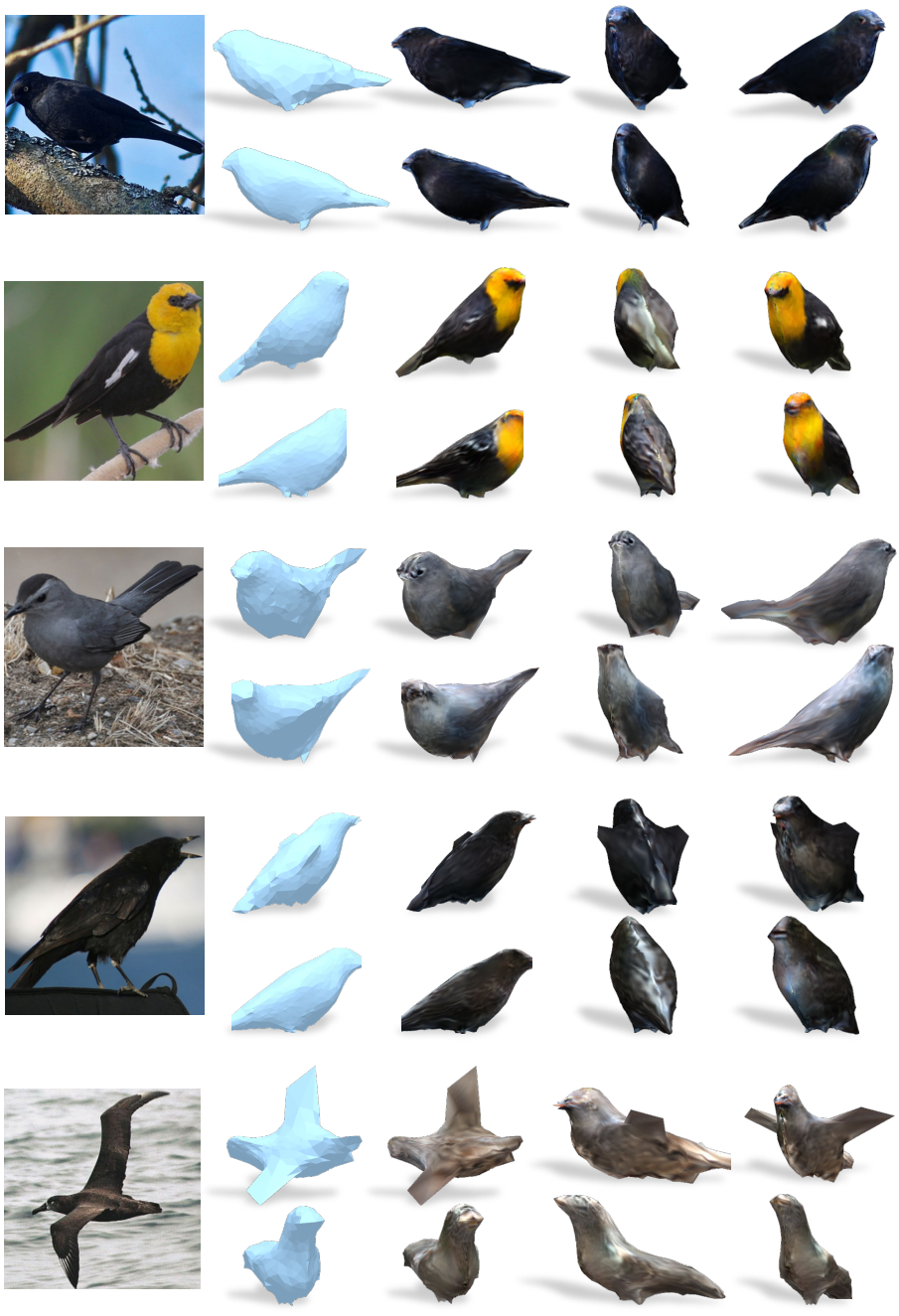

Fig. 7: **CMR-ours vs U-CMR on *random* subset.** We compare U-CMR (second row) to CMR-ours (first row) on a random subset of images from the testset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.

Fig. 8: **CMR-ours vs U-CMR on *random* subset.** We compare U-CMR (second row) to CMR-ours (first row) on a random subset of images from the test dataset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.

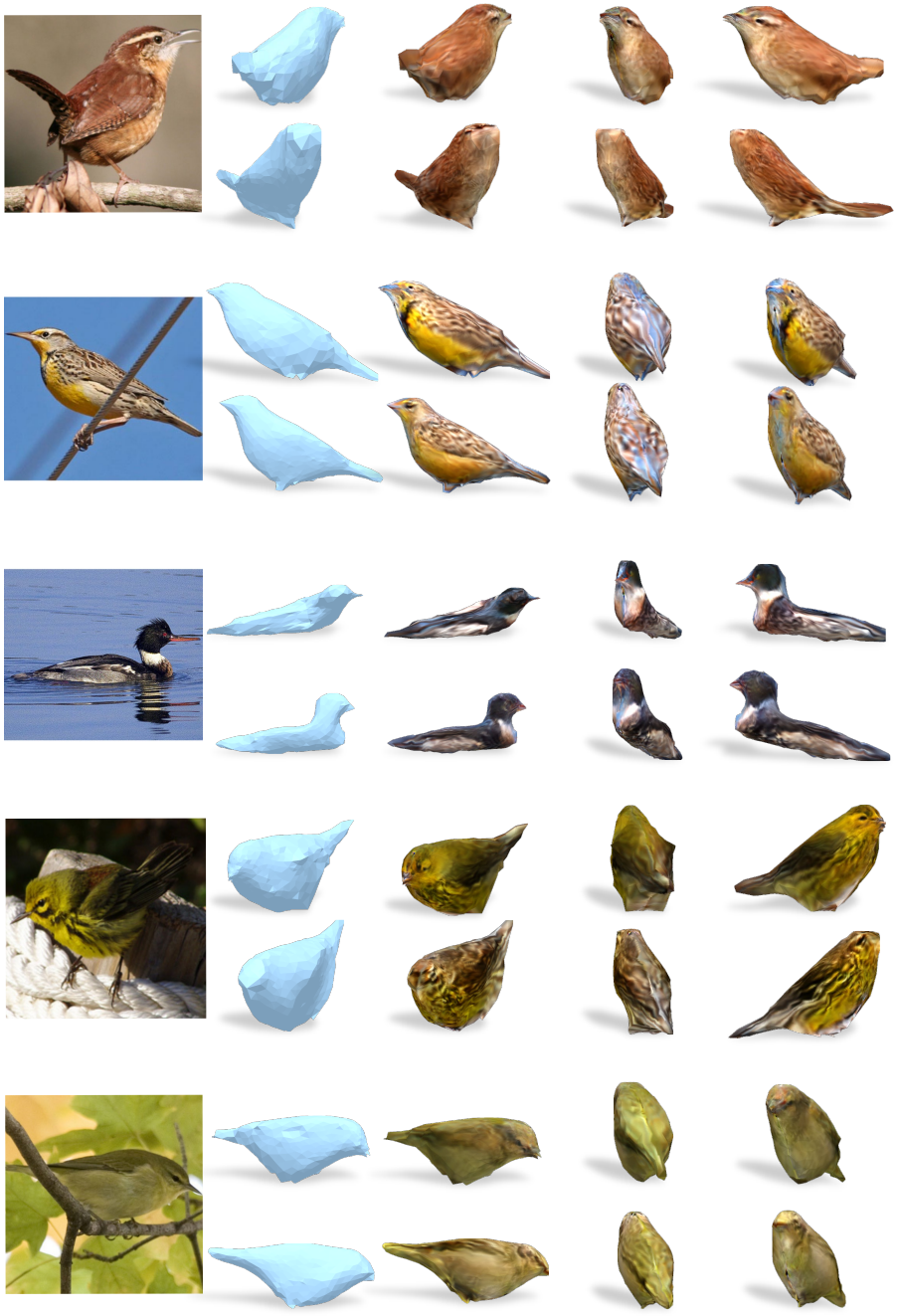

Fig. 9: **CMR-ours vs U-CMR on *random* subset.** We compare U-CMR (second row) to CMR-ours (first row) on a random subset of images from the test dataset. The first 2 columns show the predicted shape and texture from the predicted camera viewpoint. The last 2 columns are novel viewpoints of the textured mesh.

References

1. Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections. In: ECCV (2018)
2. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: CVPR (2018)
3. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In: ICCV (2019)